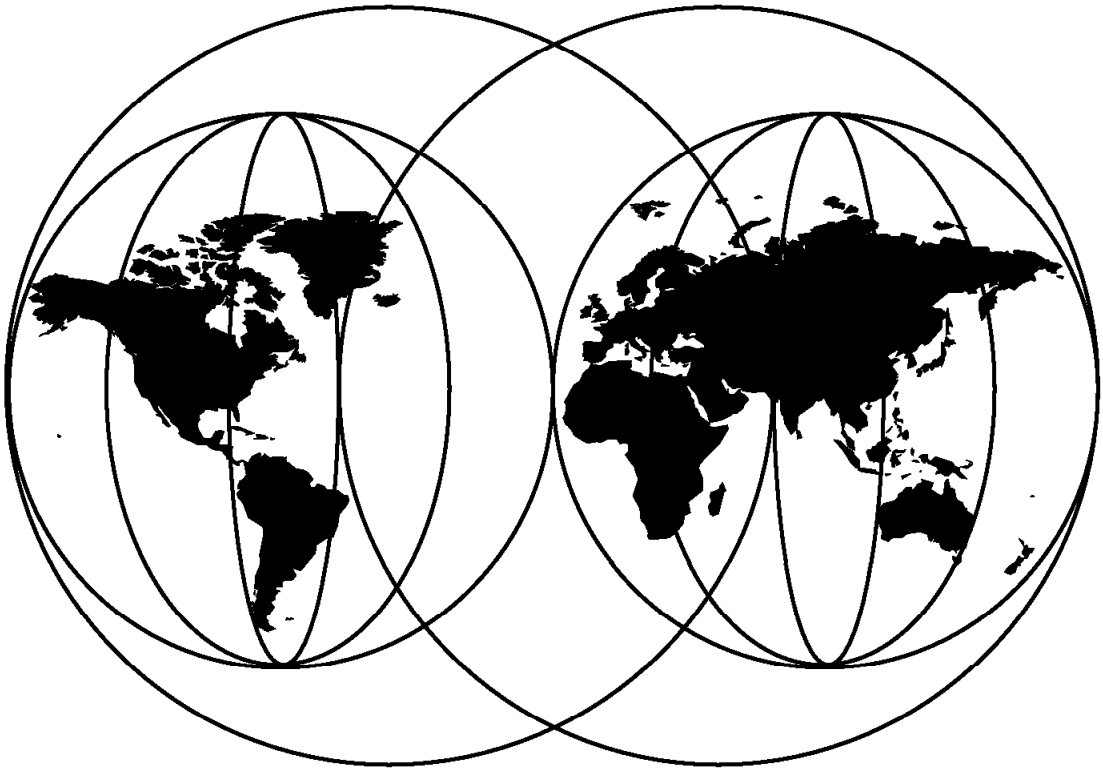
VSE to OS/390 Migration Workbook

*Cliff Bays ** Dave Greenough ** John Hutchinson*
*Dan Janda ** Kevin Jones ** Gilbert Saint-flour*



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-2043-00

International Technical Support Organization

SG24-2043-00

VSE to OS/390 Migration Workbook

October 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special Notices” on page 553.

First Edition (October 1998)

This edition applies to Version 2 Release 3 of IBM Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA), Program Number 5690-VSE, and to all subsequent releases and modifications until otherwise indicated in new editions. It also applies to Version 2 Release 4 of OS/390 (5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Contents

Figures

Tables

Preface

The purpose of this document is to provide information and guidance to personnel involved in a VSE to OS/390 operating system change; that is, a VSE to OS/390 migration.

The primary focus is on VSE program and file conversions, and on operational differences between the two systems. Chapters on each of the source languages are included. DB/DC conversions, and operational differences between POWER and JES2 are also addressed.

Within each chapter, not only are the differences pointed out, but OS/390 implementation and suggested use recommendations are made wherever possible. These recommendations can help the migrating customer ″better″ design their use of OS/390.

Throughout this document, the term MIGRATION refers to the entire process of transition from a VSE environment to an OS/390 environment. The term CONVERSION describes the process of translating and updating VSE applications and data to meet the requirements of OS/390.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working with the International Technical Support Organization Poughkeepsie Center.

Our thanks to Judith Jay for bringing a renewed focus to the issues, concerns and effort required to migrate from VSE to OS/390.

Redbook Builders and Key Contributors

Cliff Bays IBM, Endicott
Bimshire Davis IBM, Chicago
Don Durand IBM, Poughkeepsie
Dan Ebaugh IBM, Gaithersburg
Patrick Fournier Managing Partner, Automated Migration Services, Walnut Creek, CA
Dave Greenough IBM, Vermont
John Hutchinson IBM, Gaithersburg
Dan Janda IBM, Endicott
Judith Jay IBM, White Plains
Kevin Jones IBM, Endicott
Herbert Kratzer IBM, Germany
Tom Plunkett Senior Director of Systems Engineering, Automatic Data Processing, Inc., Roseland, NJ
Gilbert Saint-flour Technical Manager, Automated Migration Services, Livingston, NJ
John Sutera IBM, Endicott
Guenter Weigelt IBM, Germany

Authors and Significant Contributors

Riaz Ahmad IBM, Gaithersburg
Boris Barth IBM, Germany
Bette Brody IBM, Gaithersburg
Jerzy Buczak IBM, Cary
Charlie Burger IBM, San Jose
John Casey IBM, Dallas
Walt Farrell IBM, Poughkeepsie
Steve Gracin IBM, Endicott
Judson Howard IBM, Los Angeles
Stanley Jones IBM, Endicott
Bill Keene IBM, Dallas
Ulrich Kettner IBM, Germany
Bob Leicht IBM, Endicott
Richard Lewis IBM, Gaithersburg
Jim McCoy IBM, Gaithersburg
Tom Murphy IBM, Endicott
Karl Pesendorfer IBM, Vienna, Austria
Dave Pilcher IBM, Boulder
Linda Richter IBM, Poughkeepsie
Bernd Rueckert IBM, Germany
Liz Rushton IBM, Sydney, Australia
Roger Smith IBM, Poughkeepsie
Howard Turetzky IBM, Boulder
Jon vonWolfersdorf IBM, Endicott
Frank Yaeger IBM, San Jose
Holly Yamamoto-Smith IBM, San Jose

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 593 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Part 1. Planning the Migration - An Introduction

Chapter 1. Why Customers Migrate

This chapter discusses the following topics:

1.2, Traditional Reasons for Migrating

1.3, Functional Reasons for Migrating to OS/390

1.1 A Synopsis of This Book

What do I need to read?

Executives: Read the following:

- Part 1, “Planning the Migration - An Introduction” on page 1
- Part 8, “Migration Experience” on page 527

System Programmers: Read the following:

- Part 1, “Planning the Migration - An Introduction” on page 1
- Part 2, “Converting the VSE Operating System to the OS/390 Operating System” on page 67
- Part 4, “Converting VSE Utilities to OS/390 Utilities” on page 373
- Part 5, “Setting Up the Migration Environment” on page 399
- Part 6, “Running Your OS/390 System” on page 435

Operators: Read the following:

- Part 6, “Running Your OS/390 System” on page 435

Application Programmers: Read the following:

- Part 3, “Converting VSE Languages to OS/390 Languages” on page 247
- Part 7, “Converting your Applications” on page 479

This document is divided into nine parts:

- Part 1, Planning the Migration - An Introduction

  The scope of effort required to migrate from VSE to OS/390 will vary from one organization to another. Many factors must be considered when making the decision of when and how to migrate. This part discusses the reasons for migrating, factors to consider when sizing the effort, and developing a migration plan.

- Part 2, Converting the VSE Operating System to the OS/390 Operating System

  In this part the conversion of the VSE system including JCL, data storage methods, CICS, ICCF, telecommunications, spooling, and printing is discussed. Additionally, a comparison of the use of CMS and TSO is presented for those currently running VSE under VM.

- Part 3, Converting VSE Languages to OS/390 Languages

  Conversion of the various language compilers to their equivalent OS/390 language is discussed in this part. Also, any execution time differences are discussed.

- Part 4, Converting VSE Utilities to OS/390 Utilities

  Conversion of the VSE utilities to their equivalent OS/390 utilities is discussed in this part.

- Part 5, Setting Up the Migration Environment

  No two Information Processing environments are alike. Hardware, software, scheduling, personnel needs will be different in all cases. This part discusses preparing for and tailoring the test environment, and various hardware/software combinations and activities that can be performed in parallel.

- Part 6, Running Your OS/390 System

  The OS/390 environment is much different than the VSE environment. This part provides an orientation to the use of TSO/ISPF, OS/390 console operation, and OS/390 utilities. Additionally, the systems management philosophy with OS/390 and diagnosing problems with OS/390 are discussed.

- Part 7, Converting your Applications

  This part discusses the application program conversion process and some of the conversion tools available.

- Part 8, Migration Experience

  An example of a migration plan for the ABC company is discussed in this part.

- Part 9, Appendixes

  The appendixes provide useful information including a list of helpful publications, education information, and a chart mapping Independent Software Vendor products to OS/390 products.

1.2 Traditional Reasons for Migrating

Users migrating to MVS and OS/390 over the years have done so for a variety of reasons. While the purpose of this document is to concentrate on the hows of migrating and not so much the whys, it is interesting to note some of the more typical or traditional reasons that customers migrate to OS/390.

1.2.1 Business Consolidation

Corporations, more recently, have found themselves involved in business consolidation activities. Be it for economic and/or efficiency reasons companies have been faced with the challenge of effectively addressing this type of change. Consolidating the Information Technology infrastructure is just one of these challenges. Many have found that combining the system workloads from various parts of the newly consolidated organization has produced I/T system requirements beyond the capacity of the VSE operating system. For example, attempting to combine multiple VSE images into a single system image has often created situations where multiple processor (n-way) capacity is needed. Prior to the Turbo Dispatcher (n-way processor support) in VSE/ESA V2, OS/390 (or MVS/ESA) provided the only solution. Another issue associated with combining multiple images into a single system image has been the number of VSE partitions. Similar to the case of the Turbo Dispatcher, prior to dynamic partitions in VSE/ESA V1, OS/390 (or MVS/ESA) provided a solution to this issue.

1.2.2 Mergers/Acquisitions

As with corporate consolidations, mergers and acquisitions present an equal number of challenges when having to incorporate the I/S organizations of the companies involved. A challenge that clearly presents itself is when the organizations involved run different host based operating systems (such as OS/390 and VSE/ESA). In cases where it has been decided to merge the I/S organizations rather than run as autonomous entities, the issue of which operating system should become the single production operating system arises. It is often decided that because of its robust/enhanced functionality the operating system be OS/390. This, then, requires that the VSE subsystems and applications be converted to OS/390.

1.2.3 Capacity Constraints

Users running DOS/VSE and/or VSE/SP encountered system capacity constraints due to the architectural design limits imposed by VSE. The need for additional system capacity and resources due to things such as application and end user growth found many VSE users coming up against these constraints. OS/390 provided the much needed relief for users who found themselves in this situation. Fortunately, with the introduction of VSE/ESA V1 many of these constraints were removed.

VSE users now find that many of the reasons, due to architectural limits, that forced a conversion to OS/390 actually no longer exist. The following sections describe some of these constraints in greater detail.

1.2.3.1 Virtual Storage

VSE/SP provided 24-bit addressing which supported 16 megabytes of virtual storage. Users with the requirement for a large CICS partition, for example, were forced to go to multiple CICS partitions when putting up a single large CICS partition was not possible. This sometimes caused additional problems as it was often difficult to split a single CICS application into multiple CICS partitions. However, where possible, users chose to implement multiple CICS regions using the CICS Multiple Region Option (MRO). Still, with the addition of multiple CICS regions (MROs), comes the added expense of managing the MROs. And, as the MROs numbers increase, you need system management tools, such as CICSPlex System Manager for MVS/ESA (CICSPlex SM) to ease the system management burden caused by multiple CICS systems.

MVS, or OS/390, provided users with virtual storage constraint relief through 31-bit addressing capabilities. However, some users found relief with virtual address extensions (VAE) in VSE/SP V3. VSE/ESA V1 introduced 31-bit addressing support. This now gives VSE users the ability to address up to 2GB of virtual storage. Hence, it is now possible for VSE users with large CICS partition requirements to have this requirement satisfied by VSE.

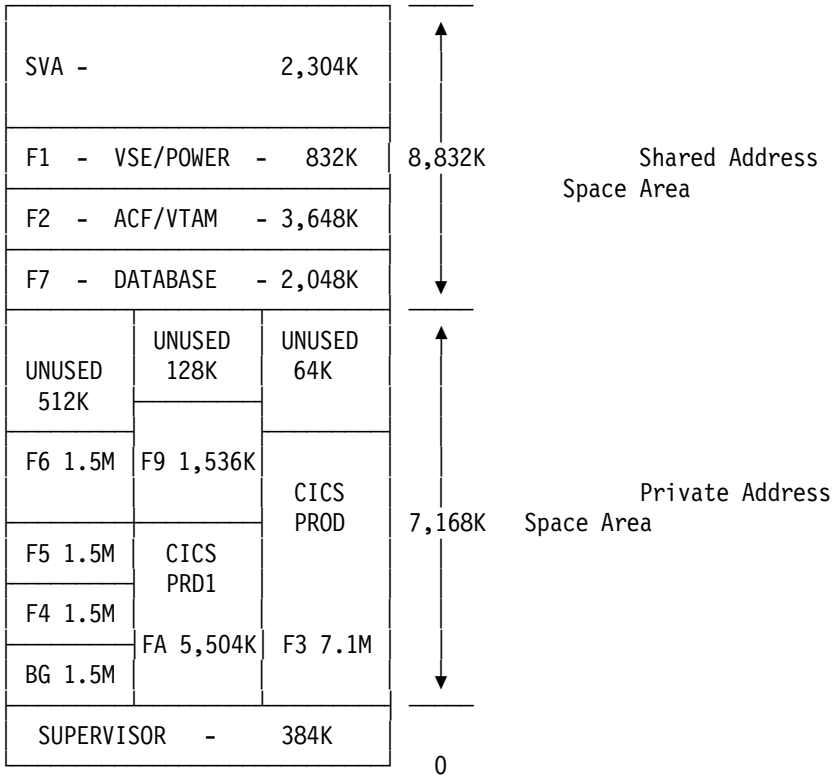


Figure 1. VAE with Three Address Spaces

Figure 1 depicts a typical VSE virtual storage configuration using Virtual Addressability Extension (VAE) introduced in VSE/SP V2. In this configuration the largest possible address space is approximately 7MB. Therefore, a single partition running in its own address space is limited to 7MB. Initially support was for only three address spaces. This was later enhanced to nine.

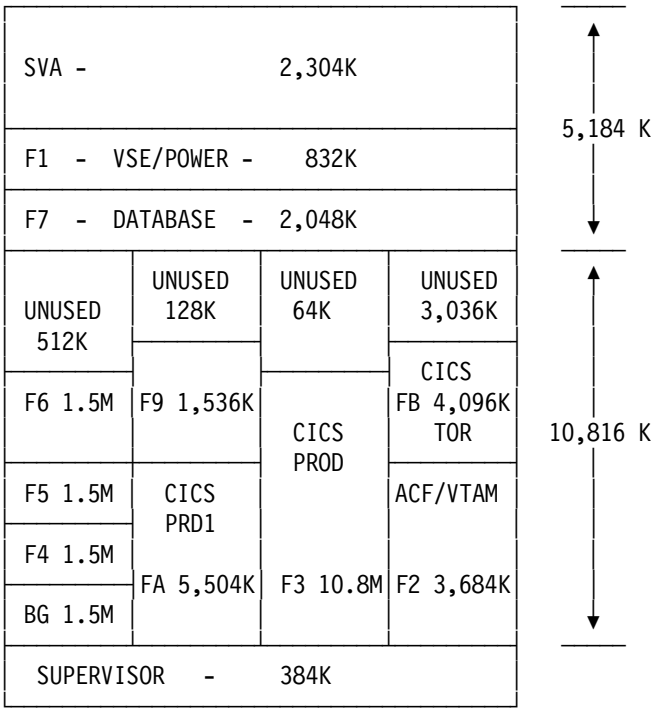


Figure 2. VAE with Four Address Spaces

Figure 2 is an example of how a customer would relieve the limitation of a 7MB private address space as depicted in the previous diagram. The 7MB limitation results from cross-system functions (for example, POWER and VTAM) having to reside in the shared area. Shared area requirements reduce the amount of virtual storage available for private area address spaces. In the above example ACF/VTAM is moved to a private address space from the shared area. This results in an additional 3.5MB for the private area address spaces. When VTAM is moved it was also necessary to move any VTAM applications into the same address space as VTAM. In this instance customers would run a CICS Terminal Owning Region (*TOR*) in the same address space with VTAM. The CICS TOR would then communicate with one or multiple CICS Application Owning Regions (*AOR*s) running in another address space. The CICS AOR was often the reason for additional private area virtual storage.

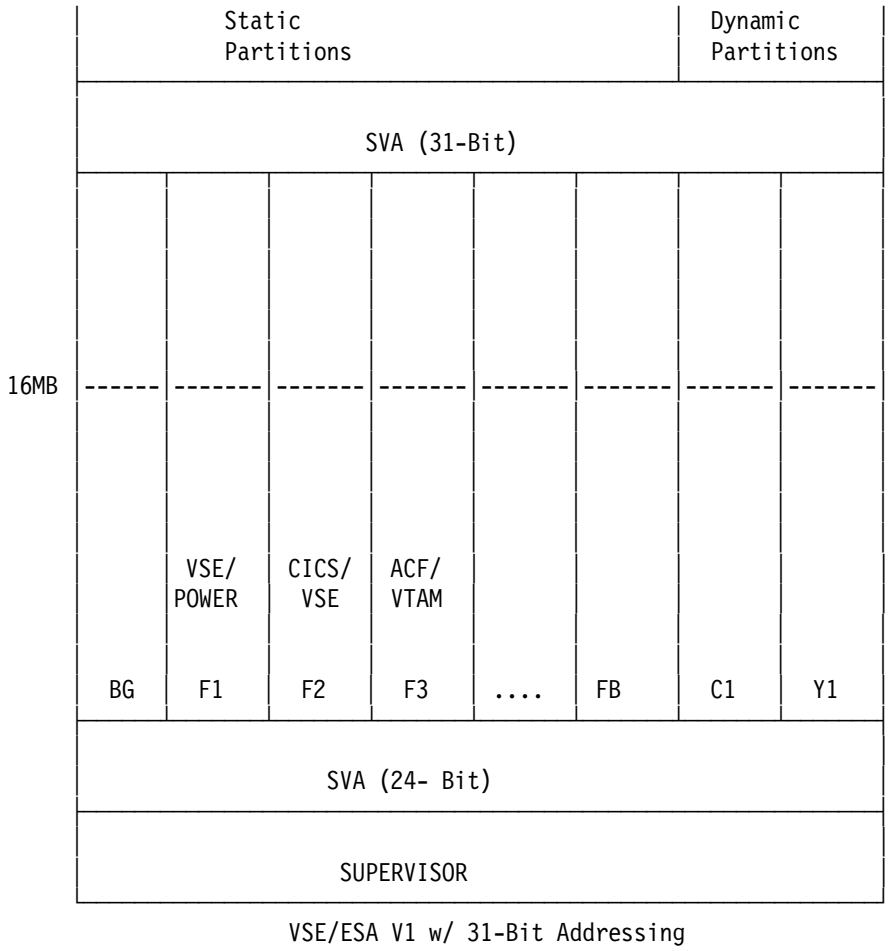


Figure 3. VSE/ESA Storage Layout

Figure 3 shows the virtual storage layout with VSE/ESA V1 or V2 exploiting Enterprise Systems Architecture (ESA) and 31-bit addressing. VSE/ESA V1, with 31-bit virtual (and real) addressing support, provides virtual storage constraint relief by extending the addressable area within a virtual address space from 16MB up to 2GB. This is a significant amount of constraint relief for both online and batch applications running in either static or dynamic partitions.

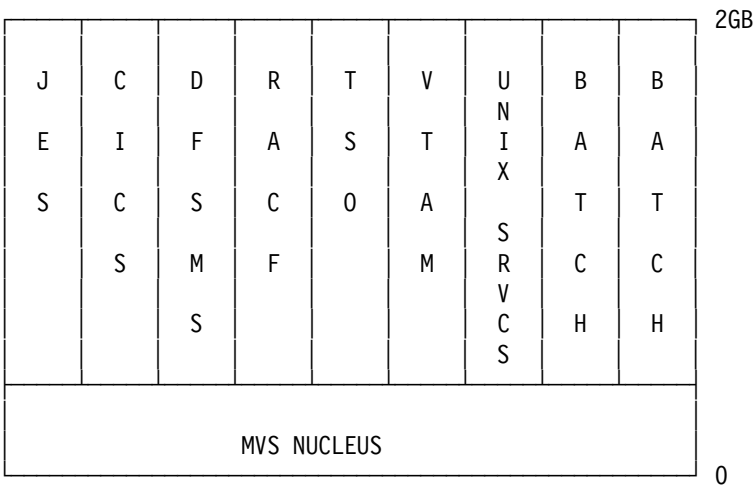


Figure 4. OS/390 Storage Layout

Figure 4 depicts a typical OS/390 system including the various functional subsystems, each running in its own address space. As in VSE/ESA, each address space has the ability to address up to 2GB of virtual storage.

1.2.3.2 N-way Processor Support

VSE/SP did not provide support for multiple processors (that is, n-way machines). Users, for a variety of reasons, exceeding the capacity of a single engine (Uni-processor) found it necessary to convert to OS/390 for its multiprocessor support. As was mentioned with virtual storage, typically these were users with a requirement for multiple CICS and batch partitions. The Turbo Dispatcher support in VSE/ESA V2 provided support for n-way processors. However, in the current version of VSE/ESA V2.2 a practical limit of only being able to support a 3-way processor exists. The number of parallel and non-parallel tasks that exist within the system workload will determine the actual number of processors that can be effectively utilized.

1.2.3.3 Task Quantity

As was mentioned in the case of business consolidations, task quantity relates to the amount of concurrent work in the system. In the consolidations example several system images (workloads) are combined into a single system image. As was also mentioned, the previous VSE system limit of 12 partitions severely limited the ability to run very large workloads; particularly those consolidated workloads requiring more that 12 partitions. The solution was to run multiple VSE images. This often created issues of managing multiple images, or deciding to migrate to OS/390.

1.2.4 Image

One final reason that users have decided to make the conversion to OS/390 is that of image. This particular reason is little talked about because it is used the least. But, it is felt that it should at least be mentioned or acknowledged.

It has been felt by some users in the VSE community that VSE is the orphan of the S/390 operating systems, ranking behind OS/390 and VM/ESA. These concerns are partly justifiable and stem from the fact that VSE has often lacked functionality provided in OS/390 and VM/ESA. Even when VSE has provided such functionality it has not done so, at least from a user perspective, in a timely

manner. That is, not concurrent with OS/390 and/or VM/ESA. This has lead users to ponder whether VSE is a viable and strategic S/390 operating system. This lack of confidence has forced these users to look at OS/390 as a more stable and strategic operating system with a viable long term outlook. An outlook that often catches the eye of upper I/S management and spurs the move toward OS/390. The introduction of VSE/ESA′s exploitation of the ESA/390 platform however has alleviated some of this doubt. It is fair to say that the focus for VSE/ESA is support for the entry to medium sized enterprises. With this in mind, it is reasonable to not expect the full array of functionality and support with VSE/ESA that one would expect with OS/390. OS/390 will continue to focus on the intermediate-large to very large ′leading-edge′ environments.

1.3 Functional Reasons for Migrating to OS/390

Besides some of the traditional reasons discussed in the previous section, there also exist some functional or other practical reasons for migrating to OS/390. While there are probably other functional reasons for migrating, this section will cover those that are typically the most common. Particularly, those that relate to applications and systems management.

1.3.1 Applications Availability

The backbone and primary purpose of any information system is its applications. This software, often considered mission critical, justifies the whole existence of information systems. Mission critical applications are those applications that are seen as most vital and crucial to the running of any business. The choice of application software is driven by business requirements. The hardware and software platforms required to support a given application is a secondary decision. Thus VSE users may find that the choice of a particular application may require the installation of another hardware and/or software platform. They may also consider a complete migration to this other software platform. Some S/390 business applications may only support OS/390. These applications may take advantage of some of the unique characteristics of OS/390 and/or its subsystems such as CICS Transaction Server, DB/2 or CICSPlex System Manager. A set of applications requiring a full function (level 2) message queuing manager, as provided by MQSeries for OS/390, is another example of OS/390 unique application capabilities.

With the announcement of Open Edition support in OS/390 a whole new set of application functions are now available to the S/390 user. Specifically, applications that were formally only available on UNIX type platforms are now available to the S/390 user. Applications such as Lotus Domino, PeopleSoft, SAP and full function Web serving bring additional application capability to the S/390 platform under OS/390.

1.3.2 Systems Management

Some of the traditional OS/390 strengths of high availability, systems management, performance, scalability and capacity have also been great attractions for the VSE user. OS/390 provides management capabilities that allow the **system** to more effectively manage the workload over those capabilities provided by VSE. Facilities such as OS/390 performance groups and the Workload manager provide this greater workload management flexibility.

OS/390 systems managed storage (DFSMS) provide enhanced system resource allocation and management. The Hierarchical Storage Manager (HSM), Removable Media Manager (RMM) and basic storage device allocation of OS/390 provide functions not inherent in the VSE environment. However, some of these functions are available from independent software vendor (ISV) products.

1.3.3 Connectivity

Connectivity, that is the ability to connect to other systems, has been one of those areas where VSE support has lagged behind OS/390 and VM. For example ACF/VTAM support for channel-to-channel connections between host systems was not introduced until VSE/AF 2.1.3. Lack of other connectivity support, that is VTAM APPN, SNI gateway, full function TCP/IP and OSA-2, has only added to the reasons why VSE users have decided to migrate to OS/390. However, as mentioned, VSE/ESA has since provided support for some of these capabilities, namely OSA-2 and VTAM APPN. VSE now enjoys virtually all of the same communications and connectivity capabilities as OS/390 and VM.

1.3.4 Systems Availability

Systems availability has always been a strong requirement for many information systems environments. Hardware and software technology enhancements in both VSE and OS/390 have brought about increased system availability. OS/390, however, has had at its core key design elements that give it premier system availability characteristics. Advanced S/390 hardware features coupled with OS/390 software functions give it this outstanding capability. VSE users have found the attractiveness of this enhanced systems availability capability, along with other features, yet another reason to embark on an OS/390 migration.

An example of OS/390 enhanced systems availability is the 3990 Concurrent Copy function when used along with BWO (Backup-While-Open) by DFSMSdss which allows backups to be taken with integrity even when control area and control interval splits and data set additions (new extents or add-to-end) are occurring for VSAM key sequenced data sets. Backup-while-open for CICS VSAM files supports SMS managed data sets without the need to close a CICS VSAM data set or to bring applications down to back up VSAM data sets. This support for backing up VSAM files while open for update is provided in conjunction with MVS Data Facility Product (MVS/DFP).

In the Parallel Sysplex environment concurrent coupling link maintenance allows the replacement of a failing coupling link without powering the CEC down. With DB2 for OS/390 copies of DB2 tablespaces can be made using DFSMS concurrent copy, a process that significantly improves data availability by reducing the time necessary to complete logically consistent copies of mission-critical data.

1.3.5 Staff Availability

In recent years S/390 application and system programming resources have become increasingly more difficult to acquire. This is particularly true for the VSE user. As the current information systems curriculums focus on the more current technologies, the traditional VSE system programming, and to some degree OS/390, skills are not being replenished. This coupled with the current high demand for year 2000 programming resources has only added to the pile of reasons that VSE users migrate to OS/390. While some amount of VSE skills are transferable to OS/390, focus is placed on developing JCL and operational skills.

Chapter 2. Sizing the Effort

This chapter discusses the following topics:

2.1 Introduction to Sizing

When undertaking a project such as migrating from VSE to OS/390 attention always turns to how much effort is really involved. The sizing effort attempts to get a fairly reasonable handle on the amount of effort and resources needed for such a project. It is desired to be able to estimate with some degree of confidence the human, system and financial resource requirements. This chapter will discuss some of the key migration activities and issues, highlighting the considerations that will affect the scope and size of this project. We will first define two terms that are often used throughout this publication, migration and conversion.

Migration is the process which takes the data processing workload, and operations from the VSE environment to the OS/390 environment. This includes a planning phase, a preparation phase, a conversion phase and a production implementation phase.

Conversion is the process within the migration where programs, data, and JCL are converted, tested, and cut over to production in the OS/390 environment.

2.1.1 Defining the Migration Project Objectives

Typical migration project objectives for an OS/390 migration project include a combination of operational needs and cost/benefit requirements.

- End-user transparency
- Minimal disruption of operations and applications support
- No overlap of dual VSE and OS/390 operations
- Standardized and automated OS/390 applications fit for automated OS/390 operations

2.1.2 Areas of VSE and OS/390 Differences

In order to properly assess and size the magnitude of the migration project, it is first necessary to understand some of the basic differences between the two operating systems. Once these differences are understood a realistic or more reasonable project outlook can be determined. The purpose of this section is to put into perspective these differences.

Even though both VSE and OS/390 support the IBM S/390 architecture, there are differences that must be considered at both the subsystem and application program level. When migrating or converting application programs from VSE to OS/390 it is important to identify these differences. The primary differences can be categorized as follows:

1. Source Programs
2. Job Control Language (JCL)
3. Files
4. Operations

2.1.2.1 Source Programs

The significance of the differences when dealing with program source code can vary by many factors. The primary determining factors involved in converting source programs have to do with the interfaces which provide services to the application programs. These application interfaces and corresponding protocols for requesting supervisor services are different in VSE than in OS/390.

The factors involved in converting batch programs that interface directly to the control program and programs that interface with application subsystems are different. Consequently, the effort and the techniques used will vary.

Source Program Inventory

The first step in assessing the scope of any application program conversion is determining the whereabouts of all of the program source code. This task must not be overlooked and needs to be done early in the conversion project. You will need to determine that all executable modules have associated source code and that all source code has associated executable modules. Executable modules missing source code, for example, will have to somehow be recreated or alternate plans developed to provide the program function. Conversion tools are available to assist in this task and are discussed later in this publication. Customers who have completed or are in the process of Year 2000 compliance are most likely aware of this issue.

The impact of source program conversion can be reduced by positioning the VSE production system with source programs compatible with both VSE and OS/390. For example, moving to the Language Environment for VSE will provide language compiler compatibility (for COBOL , PL/I and C for VSE/ESA) between VSE and OS/390.

Batch and Online Program Conversion

The conversion of batch applications must take into account differences in the application interfaces provided by VSE and OS/390. The significance of the changes required in the source programs depends a great deal on the source program language and to some extent the I/O access methods used. This

document is organized by source language type and goes into great detail at that level and includes the I/O considerations.

The conversion of the CICS applications consists of two steps. First, the VSE version of the CICS application subsystem is replaced with an OS/390 version. The two different versions of CICS contain the interfaces to the respective control programs. The second step deals with the application source code itself.

In general, the interfaces provided to the applications by the two versions of CICS are the same, the source programs do not change and need only to be recompiled with a corresponding OS/390 compiler. However, consideration should also be given to the fact that certain application level interfaces available in VSE may not be available in OS/390. The macro level API is one example. Applications written with this interface will have to be changed to use the command level API. Any access to system level control blocks should also be reviewed. Additional considerations will be required if the CICS application programs are interfacing with more than the CICS subsystem. Also, there are some source language restrictions. This document contains a section describing the CICS, DB2 and DL/I subsystems in great detail.

In summary, when comparing online and batch programs, the effort required to convert batch applications is much greater than online applications using application subsystems such as CICS and DL/I. By using application subsystems the differences in control program application interfaces become transparent to the application programmer. The installation only needs to be concerned with the common interface provided by the subsystem in situations where a VSE version and an OS/390 version are both available.

2.1.2.2 Job Control Language

All VSE JCL must be converted to OS/390 JCL. Because VSE and OS/390 differ significantly in JCL structure and syntax, this is normally one of the most complex tasks of any migration. As in the case of batch and online source programs, the considerations are more significant with the batch applications. There are, however, aids available to reduce the effort required.

2.1.2.3 Files

The impact of file conversion can be reduced by positioning the VSE production system with file formats and access methods that are compatible with both VSE and OS/390.

VSAM files are generally compatible files. One section of this document is dedicated exclusively to VSAM files and VSAM catalogs. Additional information regarding VSAM file considerations can be found in the different source language sections.

Direct Access Method (DAM) files require individual evaluation because each can have unique characteristics. Each of the language sections has a description on accessing DAM files. It is recommended that these file structures be converted to relative record VSAM files where possible.

Sequential tape files are compatible between VSE and OS/390. There can be differences in the format of the labels and how they are processed. There is a chapter in this publication that deals exclusively with tape files.

Sequential DASD files are compatible between VSE and OS/390. However, OS/390 does not support sequential (SAM) files located within VSAM managed space. These files will have to be reloaded to different DASD areas before OS/390 can process them.

DL/I databases are compatible with IMS databases [1] if the "IMSCOMP" parameter was specified during the DL/I DBD generation. If this parameter was not specified, then reloading of the database will be necessary; that is, a VSE positioning activity. Similarly, DB2 for OS/390 provides compatibility with DB2 for VSE. A section on database differences is included in this publication.

Note: *VSE DASD volumes can be read and processed by OS/390. VSE DASD volumes, when first read by an OS/390 system, will have their free space areas calculated and appropriate entries recorded in the volume's VTOC. VSE systems can later process these volumes as required. (Even though OS/390 has written new records in the VTOC, VSE will ignore them.) Never, however, should both systems have concurrent access to DASD volumes. Also, all volumes are required by OS/390 to have unique volume serial numbers.*

2.1.2.4 Operations

OS/390 operational procedures and operator commands differ significantly from those used in VSE. The input/output spooling subsystems (VSE/POWER and OS/390 JES) are quite different in function and operations also. Some of these differences are addressed in this publication.

Changing operational procedures and training of operators in the operations of OS/390 are very important tasks that must be performed during a VSE to OS/390 migration. Training courses are available to assist in this effort.

2.1.3 Comparison of Basic VSE Functions & Components to OS/390

Here is a list of some of the areas or programs that may be affected:

Table 1 (Page 1 of 3). Comparison of VSE Functions & Components to OS/390 Replacements

VSE	OS/390	Comment and Reference
VSE Base Functions IOCP POWER (w/PNET, RJE) EREP MSHP ...	OS/390 Base Functions IOCP, HCD * JES2, JES3 (w/NJE, RJE) * EREP * SMP/E * ...	
Application Generators VisualAge VisualLift SDF/CICS & SDF II CSP DMS/CICS	Application Generators VisualAge VisualLift * SDF II CSP DMS/CICS	

[1] IMS databases are the OS/390 equivalent of VSE DL/I databases.

Table 1 (Page 2 of 3). Comparison of VSE Functions & Components to OS/390 Replacements

VSE	OS/390	Comment and Reference
Languages	Languages	See Part 3 page 247
LE/VSE	LE/MVS *	Chapter 17 page 351
HLASM	HLASM *	Chapter 13 page 267
COBOL	COBOL	Chapter 12 page 249
PL/I	PL/I	Chapter 15 page 333
RPG II	RPG II	Chapter 14 page 329
REXX	REXX	Chapter 18 page 369
FORTRAN	FORTRAN	Chapter 16 page 349
C	C/C++ *	
AFP Family	AFP Family	See Chapter 11 page 235
PSF/VSE	PSF/MVS	
PPFA	PPFA	
OGL	OGL	
Font Libraries	Font Libraries	
...	...	
Network Management	eNetwork Comm. Server	See Chapter 9 page 185
VTAM (APPC, APPN)	VTAM (APPC, APPN) *	
NCP	NCP *	
BTAM/ES	BTAM/SP	
TCP/IP	TCP/IP *	
LANRES	LANRES *	
MQSeries	MQSeries	
NetView - CSF	NetView	
CICS/VSE	CICS Transaction Server	See Chapter 6 page 133
CallPath	CallPath	
DISOSS	DISOSS	
...	...	
Console Management	Console Management	See Chapter 28 page 443
OCCF	MCS & MPF*	
Console Automation (ISV)	SDSF *	
	NetView AOC	
	NetView	
Systems Management	Systems Management	See Chapter 30 page 457 TSO/ISPF Panels do not provide the JCL generation function of the Interactive Interface
MSHP	SMP/E *	
DSNX	NetView DM	
Interactive Interface	TSO/ISPF panels *	
Explore (ISV)	RMF *	
VSE/PT	RMF *	
OMEGAMON (ISV)	RMF *	
Development Environment	Development Environment	See Chapter 7 page 155 and Chapter 27 page 437
ICCF	TSO/E *	
CMS	ISPF/PDF/SCLM *	
Programming Library Mgmt:	ISPF Functions:	See Chapter 22 page 389
VSE Librarian, Panvalet (ISV)	DM, PDF, LMF, SCLM *	
	DM, PDF, LMF, SCLM *	
Security Manager	Security Server	
ACLR	RACF *	
ALERT (ISV)	RACF *	
Disk Management	DFSMS Family *	See Chapter 24 page 397
ICKDSF	ICKDSF *	
VSAM	DFSMSdfp (VSAM) *	
Dump/Restore/Fcopy	DFSMSdss *	
CA-Dynam/D (ISV)	DFSMShsm *	
Tape Manager	DFSMS Family *	
CA-Dynam/T (ISV)	DFSMSrmm *	

Table 1 (Page 3 of 3). Comparison of VSE Functions & Components to OS/390 Replacements

VSE	OS/390	Comment and Reference
ADSM/VSE	ADSM/MVS	
DITTO/ESA for VSE	DITTO/ESA for OS/390	See Chapter 20 page 381
DFSORT/VSE	DFSORT *	See Chapter 19 page 375
Data Base Management DL/I DB2 (SQL/DS) QMF	Data Base Management IMS/DB DB2 QMF	See Chapter 8 page 169
GDDM/VSE	GDDM/MVS *	See Chapter 7 page 155
Dump Analysis Info/Analysis (VP)	Dump Analysis IPCS *	
Job Scheduler (VP)	OPC/ESA	
Report Manager (VP)	RMDS	
NetView family FTP	NetView family FTP	

Key: ***(*)*** = Package is an element of OS/390. ***(ISV)*** = VSE function provided by independent Software Vendor

2.2 OS/390 Components/Products/Subsystems

Note: The terms OS/390 and MVS (including MVS/XA, MVS/SP, and MVS/ESA) may be used interchangeably throughout this publication. OS/390, with its integrated components, refers to the current version and all previous versions of MVS unless otherwise noted.

Another important aspect to consider when sizing the migration is determining which OS/390 components will be installed. This is basically determined by assessing which OS/390 components and/or optional products provide functions comparable to those in VSE. The previous table in this section provided a comparison chart for this purpose. What follows is a brief description of some of the key OS/390 components.

There was discussion about including that which is the same between the operating systems. This can be a big item when customers are also entertaining ideas of other operating systems. There is more closeness between VSE and OS/390 than with RISC6000/UNIX to OS/390. The move to UNIX will require a new start or complete rewrite.

This is a frequent consideration when customers are considering implementing ERP Applications. They can put these core business integrated packages (for example SAP R3, Bond, JDEdwards, Oracle) on either a UNIX or OS/390 platform.

2.2.1 The OS/390 Operating Environment

This section introduces the OS/390 operating environment. A publication entitled *OS/390 Introduction and Release Guide*, GC28-1725 is recommended for a better understanding of OS/390. This book describes the information associated with OS/390 including OS/390 books and books for participating elements.

2.2.1.1 OS/390 Product Content

The operating system environment that is called OS/390 consists of MVS/ESA SP and its component products and functions.

Base Elements

As an example, OS/390 Version 2 Release 5 contains the base elements listed below. Subsequent releases of OS/390 will contain similar components, their replacements.

- ***System Services***
  - MVS/ESA SP
  - Base Control Program (BCP)
  - DFSMSdfp
  - EREP
  - ESCON Director Support
  - IBM High Level Assembler for MVS
  - ICKDSF
  - ISPF
  - JES2
  - MICR/OCR Support
  - MVS/Bulk Data Transfer (BDT Base)
  - TSO/E
  - 3270 PC File Transfer Program
  - FFST
  - TIOC
- ***Systems Management***
  - HCD
  - ICSF
  - SMP/E
  - SystemView for MVS Base
- ***Application Enablement***
  - Language Environment
  - DCE Application Support
  - Encina Toolkit Executive
  - GDDM/MVS (includes PCLK and OS/2 Link)
  - OS/390 Application Enabling Technology
  - SOMobjects Runtime Library
  - VisualLift for OS/390 Runtime Library
  - C/C++ IBM Open Class Library

- ***Distributed Computing***
  - UNIX Application Services (Shell, Utilities, and Debugger)
  - UNIX System Services (included in the BCP)
- ***Distributed Computing Services***
  - DCE Base Services (OSF DCE level 1.1)
  - DCE DFS (OSF DCE 1.2.1 level)
  - DFSMS/MVS Network File System
- ***eNetwork Communications Server***
  - VTAM (includes the AnyNet function)
  - IBM TCP/IP
    - CICS Sockets
    - Host on Demand
    - IMS Sockets
    - Domain Name Server and WLM support (DNS/WLM)
- ***Network Computing Services***
  - Domino Go Webserver for OS/390
    - NetQuestion
    - Internet Connection Secure Server
  - IBM BookManager BookServer for World Wide Web
- ***UNIX System Services***
  - OS/390 UNIX System Services Application Services
  - OS/390 UNIX System Services Shell & Utilities
  - OS/390 UNIX System Services Debugger
- ***LAN Services***
  - LANRES
  - LAN Server
  - OSA Support Facility
- ***Softcopy Publications Support***
  - BookManager READ/MVS
  - Softcopy Print (includes Softcopy Print for DBCS Languages)

Optional Features

These are priced as well as unpriced features included in OS/390 integration-testing. The host-based features are capable of being dynamically enabled or disabled. As an example, here is a list of optional features for OS/390 Version 2 Release 5:

- ***System Services***
  - JES3
  - MVS/BDT File-to-File
  - MVS/BDT JES3 SNA NJE
- ***Security Server***
  - OS/390 Security Server (RACF and DCE Security Server at OSF DCE level 1.1)

- ***Systems Management Services***
  - DFSMS/MVS features (DFSMSdss, DFSMSrmm, DFSMShsm)
  - HCM
  - RMF
  - SDSF
- ***Application Enablement Services***
  - DFSORT
  - GDDM-PGF
  - GDDM-REXX/MVS
  - IBM C/C++ Compiler (with debug tool)
  - IBM C/C++ Compiler (without debug tool)
  - IBM High Level Assembler Toolkit
  - Language Environment Data Decryption
  - SOMobjects Application Development Environment
  - VisualLift Application Development Environment for MVS, VSE, and VM.
- ***Distributed Computing Services***
  - DCE User Data Privacy (DES and CDMF) - OSF DCE 1.1 level
  - DCE User Data Privacy (CDMF) - OSF DCE 1.1 level
  - OS/390 Print Server (includes IP PrintWay/NetSpool)
- ***eNetwork Communications Server***
  - IBM TCP/IP Kerberos (DES)
  - IBM TCP/IP Kerberos (non-DES)
  - IBM TCP/IP Network Print Facility
- ***Network Computing Services***
  - Domino Go Webserver for OS/390 Export Security Feature
  - Domino Go Webserver for OS/390 North America Secure Feature
- ***Softcopy Services***
  - BookManager BUILD/MVS

See the latest version of the *OS/390 Introduction and Release Guide* for an up-to-date list of OS/390 features and complete descriptions.

2.2.1.2 MVS Subsystem and Component Terminology

The following are some of the major subsystem programs or functional support facilities that are provided as integral components of an OS/390 system. They can help an installation manage and control their MVS operating system environment.

- ***Job Entry Subsystem/2 (JES2)***

  One of two MVS system facilities for spooling, job queueing, and managing input and output. This publication will address VSE POWER to MVS JES2 migrations.

- ***Job Entry Subsystem/3 (JES3)***

  The second MVS system facility for spooling, job queueing, and managing input and output. JES3 will not be addressed in this publication.

- ***Data Facility Storage Management Subsystem***

  Complementary functions of MVS/DFP and other individual products of the Data Facility family which, together with RACF, provide a system-managed, administrator-controlled storage environment.

- ***Systems Resources Manager (SRM)***

  A system function that determines which address spaces should be given access to system resources (for example processor, storage, I/O), and the rate at which each address space is allowed to consume resources. To a large degree, an installation′s control over the system is exercised through the SRM; that is, via SRM tuning parameters.

- ***Systems Management Facility (SMF)***

  A facility that gathers and records job accounting and other system-related information. By creating analysis and report routines, the collected information can be used for billing users, for analyzing workloads, and for profiling system resource usage.

- ***Interactive Storage Management Facility (ISMF)***

  An interactive, online facility for defining and viewing the policy of how the Storage Management Subsystem manages auxiliary storage.

- ***Time Sharing Option Extensions (TSO/E)***

  TSO/E provides interactive time sharing capabilities.

- ***Interactive System Productivity Facility (ISPF)***

  Dialog manager required for interactive applications; for example ISPF/PDF, ISMF, and IPCS sessions.

- ***Interactive System Productivity Facility/Program Development Facility (ISPF/PDF)***

  ISPF/PDF provides enhanced edit and browse facilities for aiding program development and library management functions.

- ***Integrated Catalog Facility (ICF)***

  The name of the catalog in DFP that is a functional replacement for VSAM catalogs.

- ***Interactive Problem Control System (IPCS)***

  An interactive, online facility used for diagnosing software failures; that is, dump viewing.

- ***Message Processing Facility (MPF)***

  The facility that controls console message processing and message display. Message processing refers to message suppression, message retention, and the use of installation-supplied exits to control message processing.

- ***Global Resource Serialization (GRS)***

  A component of MVS designed to protect the integrity of resources, particularly data sets on DASD volumes that are shared by two or more systems.

- ***System Modification Program Extended (SMP/E)***

  SMP/E controls software changes to modules and macros in the operating system, using a standard format and method that help ensure system integrity. SMP/E is required for installation and service functions.

2.2.1.3 Supporting Products

A typical OS/390 operating system environment also includes several other, both required and optional, system-related products.

Some of these products are described in alphabetical order below.

- ***Data Facility Data Set Services (DFDSS)***

  DFDSS copies, moves, dumps, and restores data sets and volumes for backup and recovery. It can be used to migrate data sets from one DASD device to another. It is the product used to convert data to and from the Storage Management Subsystem.

- ***Data Facility Hierarchical Storage Manager (DFHSM)***

  DFHSM backs up, recovers, and manages space on volumes.

- ***Data Facility Sort (DFSORT)***

  DFSORT sorts, merges, and copies data set records.

- ***MVS/Data Interfile Transfer, Testing, and Operations Utility (DITTO)***

  DITTO is a general-purpose utility program for tape, disk, and card input/output devices. Can be used interactively under ISPF.

- ***Device Support Facilities (ICKDSF)***

  Device Support Facilities initializes DASD volumes and recovers data from defective tracks. It can also be used to migrate to indexed VTOCs. (This is included in the base OS/390 product.)

- ***Resources Access Control Facility (RACF)***

  RACF controls access to data processing resources.

- ***Resource Measurement Facility (RMF)***

  RMF measures and reports on the performance and availability of the system.

- ***System Display and Search Facility (SDSF)***

  SDSF helps authorized users monitor and control the operation of an MVS-JES2 system. SDSF consists of online panels that provide immediate information about jobs, queues, initiators, and active tasks.

- ***TME 10 Information/Management***

  Implement, enforce, and automate administrative processes and policies in your enterprise. TME 10 Information/Management offers you an integrated platform of tools, services, and interfaces to accomplish this. In addition, TME 10 Information/Management provides a centralized repository capable of storing up to 400 Gigabyte of data per database on an MVS/ESA or OS/390 platform. It also integrates with many of Tivoli′s TME 10 (Tivoli Management Environment) software products.

There are of course more system-related products available to support OS/390 installations. The ones listed above are mentioned because of their broad applicability in many environments. Not all of those listed may have applicability

in your environment. Each should be researched individually for installation applicability.

2.2.2 Subsystem Level Comparison/Affinity

Various sections in this publication deal with the VSE and OS/390 subsystems and detail their similarities and differences. Specifically, these subsystems are:

- DB2/VSE and DB2/MVS
- DL/I and IMS/DB
- CICS/VSE and CICS/ESA
- POWER and JES
- Telecommunications (VTAM, NCP, BTAM)
- ICCF and TSO

Refer to these sections for specific details of subsystem level comparison/affinity and migration issues.

2.3 What Changes Between VSE and OS/390?

The particular items discussed in this section may have some significant impact as you enter the OS/390 environment. How it is decided to implement the changes in these key areas will effect the amount of effort and resources that will be required for the migration and subsequent production environment.

2.3.1 Philosophical Changes

One of the most signification philosophical changes when going from VSE to OS/390 is that of the design points of the two operating systems. OS/390 has at its design point a strong focus on systems management, specifically systems availability. Thousands of lines of operating system code is dedicated to preventing and/or reducing IPLs, system and program ABENDs and unscheduled downtime. This may mean a big change for those VSE environments where frequently scheduled IPLs are a regular occurrence.

2.3.1.1 Security

Customers who have developed security policies and procedures within the VSE environment will find developing similar policies and procedures under OS/390 fairly straightforward. However, as with many of the products providing equivalent VSE function in the OS/390 environment actual product implementation may be different. This applies also to the security product chosen for OS/390. OS/390 focuses on system integrity, that is, security checking is done prior to performing any function.

Customers who have chosen to implement little or no security with VSE may find themselves doing so with OS/390. If this is the case then policies and procedures will have to be developed and a security package, for example RACF, chosen to implement these policies and procedures.

2.3.1.2 Automation

VSE customers who use OCCF and/or ISV products to provide console automation functions will find enhanced function in the OS/390 environment. Because of the availability of functions such as DFSMSrmm and DFSMShsm consideration will have to be given to how to best implement these functions, starting with the development of storage and media policies. ISV products also exist in the OS/390 environment to provide additional automation capabilities.

2.3.1.3 Console Operator Interface

VSE console operators tend to have a significant amount of interaction with the system console. This can be referred as a ′chatty′ interface. Many batch applications depend upon operator responses to function correctly. For example, an operator may be required to enter date information and response verification in order for a program to continue. Such facilities are not provided in OS/390 requiring these type of applications to be redesigned.

OS/390 provides the Message Processing Facility (MPF) which controls console message processing and message display. MPF is similar in function to VSE OCCF.

2.3.1.4 JCL Processing

VSE JCL syntax and structure is very forgiving and flexible. Users often exploit this capability to enhance user productivity. For example, users often intentionally code invalid JCL statements so that they may appear on the system console for the correct information to be entered. This, then, provides a somewhat crude way of creating dynamic JCL decks. This capability exists because of the manner in which VSE POWER performs JCL processing. POWER does JCL syntax checking at job execution time. When an invalid statement is encountered the console operator is given the opportunity to enter the correct statement. OS/390, however, does not provide such a capability because of the way JES is designed. With JES, JCL syntax checking is performed at job submission time. Jobs with invalid statements are rejected at this point and, therefore, not executed. Consideration will need to be given to POWER jobstreams that are designed in such a manner.

2.3.1.5 Management Disciplines

Because of OS/390′s enhanced systems management capabilities, thought needs to be given to system management, and its various disciplines, and how it will be implemented in the OS/390 environment. OS/390 provides functions and capabilities in each of the systems management areas. Specifically these are:

- Change control
- Problem control
- Performance management
- Capacity planning
- Configuration management

2.4 Who is Affected by This Migration?

2.4.1 Job Roles and Normal Activities

The following table which lists job roles and activities is intended to link specific activities to the appropriate job role. As such, it is also intended to act as an aid in determining the impact of the migration project on the various I/S functions. For example, assigning skills development to application program development and data center operations is useful when developing the education plan for the migration project. This will take into account the timing of who will get education and when.

Table 2. Who's Normal Activities are Affected?

Roles	**Activities**										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Application Program Development		*	*	*				*			
Applications End-Users								*			
Auditor								*			
Data Center Operations	*		*					*			
Help Desk	*										
Management					*					*	*
Network Support Staff	*										
Performance Analyst						*					
Production Control	*							*			
Quality and Testing								*			
RJE End-Users & Operations	*							*			
Systems Programmers			*		*	*	*	*			

Activities

1. Procedures
   - Run Book
2. Standards
   - Coding
   - New Programs
   - Naming Conventions
3. Skills Development
4. New Tools - Application Development

5. Security
6. Performance
7. Capacity Planning
8. Testing
9. Backup/Recovery
10. Disaster Planning
11. Project Plan Development

2.5 Approaches to Migration

2.5.1 Disclaimer

For the purpose of providing a more effective guide the mass migration method was adopted as an approach or strategy in migrating. The reasons for the choice are numerous, but they include:

- Mass migration provides a project duration that is definable. This allows for a more accurate migration project cost estimation and sizing.
- In today′s integrated I/T environments it is more difficult to define discrete kernels. For example, many applications currently have integrated facilities that support the integrated nature of many business functions. This can be found in applications such as Enterprise Resource Planning (ERP). The sales forecasting function, for example, shares information with certain accounting functions. This makes it difficult to separate or define discrete kernels to migrate.

2.5.2 OS/390 Conversion and Production Implementation Strategies

There are two different strategies (or approaches) you can use in migrating applications to OS/390. They are: (1.) the **kernel/progressive** approach, and (2.) the **single switchover - mass application migration** approach. The decision as to which approach to take will have a definite impact on the project, particularly on the manner in which resources are deployed Additionally, the approach decision will, in most cases, have the greatest impact on sizing the project. The following discussion presents these two approaches.

2.5.2.1 Kernel/Progressive Approach

Here, an installation defines discrete application sets called kernels[2]. The conversion team uses progressive conversions of each defined kernel, placing a converted kernel into OS/390 production on a ″when ready,″ serial basis. After a kernel is cutover[3] to OS/390 production, the next defined kernel is worked on, converted, and implemented on OS/390. This process goes on until all applications (kernels) are cutover to the OS/390 environment. Some points to make about the ″kernel approach″:

[2] A kernel is usually defined as all the programs and files that are needed to support a business application; for example, the payroll system.

[3] ″Cutover″ is a term generally associated with the kernel approach. It is a word used to describe the completed conversion of a kernel to OS/390; that is, the time when the kernel is placed in OS/390 production.

- OS/390 production is realized at an early time in the migration.

  When the first kernel is completed it is cut over to OS/390 production. This could be at a very early time in the migration thus providing early OS/390 feedback.

  However, this may not be the advantage it appears to be. Dual OS/390 and VSE production environments exist as VSE production (of unconverted kernels) is required. This can be a disadvantage operationally as well as cause problems in resource (I/O) scheduling.

  Many times, because of the dual production environment, application bridges must be built (special procedures) to allow data and catalogs to be alternately processed by the OS/390 and then the VSE system. Also, maintenance and development activities must be performed on both systems, thus potentially slowing down the overall migration.

- Dedicated and rotating conversion teams are usually involved.

  The system programmer contingent of the conversion team is mainly dedicated to the migration effort. However, application programmers very often are involved in converting their own applications with this approach. Rotating application programmers in and out of migration efforts can be detrimental to development activities. It can also slow down overall effectiveness of the migration as additional time and training takes place each time new personnel are assigned to the conversion team.

- No definite project-end date is likely to be associated with this approach.

  Many times with the kernel approach, the conversion effort ″runs out of steam″ before the project is completed. This happens after the important bread-and-butter kernels are cutover. Then, priorities often change and the lesser visible applications stay operational under VSE for long periods of time. This becomes expensive to a company as additional resources are involved in maintaining two operating systems and managing two production environments. This is why the phrase ″it took us 18 months or two years″ is many times muttered about a VSE to OS/390 migration.

2.5.2.2 Single Switchover - Mass Application Migration Approach

In the single switchover - mass application migration approach, all applications are cutover to OS/390 production at the same time. (This time is often referred to as the ″switchover″.) As applications are converted and successfully tested under OS/390, they are ″shelved″ until switchover. [4] At switchover, VSE operations stop in entirety and OS/390 operations commence. A comprehensive conversion aid tool (that is, product) is almost always used with this approach. Some of the advantages to the single switchover - mass application migration approach are:

- OS/390 operations are deferred until project completion.

  The advantage of this is that there is no dual operations. Operators run VSE production until the conversion is over. Also, there are no special ″bridges″ that have to be built between the two systems since there is no need to move production data back and forth between VSE and OS/390 systems.

- A dedicated conversion team is usually associated with this approach.

---

[4] Maintenance updates can continue to be made to these ″converted″ VSE applications. The changes should be made to the VSE source programs. Later these programs will have to be cycled back through the conversion process.

A conversion team is normally chosen that will be dedicated to the project until its end. Included with this team will be (perhaps) two application programmers. Naturally, the number varies with the size and complexity of the project. The team is responsible for converting all VSE applications. (As previously mentioned a program conversion aid is normally used with this approach.) Application programmers, not part of the project team, are not disrupted during conversion work. They can continue to perform VSE application development and maintenance activities.

- The migration project has a visible end.

  Because the project is an important one (obviously it wouldn′t have been undertaken otherwise), and since no applications are cutover until all applications are ready, the conversion effort will not lose steam. Priorities will remain very high to complete application conversions, and to implement OS/390 as the production system. Typically, the duration of realizing total OS/390 production with this type of approach, is significantly less, (even up to 50 percent less in duration), then with the kernel approach.

- Staff is better prepared, trained, and experienced with OS/390 prior to production operations.

  OS/390 skills are developed during all conversion activities; that is, conversion activities are performed on the OS/390 system. All learning and hands-on activities are accomplished on a non-production OS/390 system, thereby lessening future production exposures. Since there is no dual operations of both VSE and OS/390, operators don′t get confused as to which system they′re operating on.

2.5.3 VM/ESA Guest Support in Your VSE to OS/390 Migration

VM/ESA′s Guest Support has long been an important part of many VSE and MVS(OS/390) customers operating environments. As you approach migrating VSE to OS/390 you should consider the important roll VM/ESA plays in making the job easier and more cost effective current and long term.

If you already have VM/ESA and you use VM/ESA′s Guest Support for running your VSE system(s) then you already know the value VM/ESA delivers in this environment. In migrating VSE to OS/390, VM/ESA continues to play an important roll delivering as much or more value to your new OS/390 environment. If you are not familiar with VM/ESA a more complete description of how to implement multiple VSE and OS/390 images can be found in chapter 26 of this publication. Chapter 26 also discusses the benefits and consequences of using VM/ESA and LPAR to support multiple images both during and after the migration. For more information on VM/ESA obtain a copy of *VM/ESA V2R2.0 General Information*, GC24-5745 and *VM/ESA V2R1.0 Running Guest Operations*, SC24-5755.

2.5.4 Staffing Strategies

2.5.4.1 In-House Staff

There are two main strategies involved when deciding how to staff the migration project. These typically are using existing in-house staff or hiring outside consultants. Some considerations when using in-house staff are:

- Staff availability

  Deciding to use in-house staff as part of the migration makes it difficult to perform regular job responsibilities while they are involved with the migration project. This is particularly true of applications staff as current application development and maintenance has to be put on hold.

- Staff Skills

  When using in-house staff basically the same education requirements exist as those for outside consultants. These requirements are usually satisfied through in-house or classroom education. However, using in-house staff for the migration project also develops migration and conversion skills. These skills, such as training on the migration method and use of any migration/conversion tools, may not be of benefit after the migration project. This may provide a reason to acquire them from an outside source.

2.5.4.2 Outside Consultants

The alternative to using in-house staff is outside consultants. As with in-house staff, using outside consultants has its considerations. Chiefly this is the fact that outside consultants already bring with them expert levels of skill and experience. One of the main benefits of exploiting this skill and experience is that it tends to shorten the duration of the project. Utilizing outside consultants also frees existing in-house staff to perform their regular job duties. It may also be desired to hire new system personnel that already possess OS/390 (MVS) skills. Lastly, one of the big considerations is the amount of financial resources that will be required to use outside consultants. The forecasted project length and number of consultants needed are obviously the major factors. There are consulting firms that specialize in migrations such as this. While IBM in no way endorses or warrants their work performance, listed below are a few of the firms that specialize in migrations:

- Automated Migration Services
- CAP-GEMINI
- IBM Global Services
- MHT Services

2.5.5 Conversion Tools

There are a number of conversion tools available to assist in the migration project. Some of the considerations when selecting conversion tools are:

- Cost
- Education requirements
- Technical support
- Effectiveness
- Flexibility

Listed are a few of these tools. A chapter in this publication on conversion tools provides detailed information about these tools.

- Program Translators (IBM CCCA)
- Emulation - (Computer Associates CA-DUO)
- Program Source Recovery - (Source Recovery)

- Mass conversion - (Cortex-MS)
- Program inventory - (IBM OPTI-AUDIT)

2.6 Educational Requirements

2.6.1 Introduction

The educational requirements for the migration project will generally take the form of developing OS/390 skills; that is, JCL and conversion techniques. With the latter, strategies will have to be developed to convert things such as VSE program source and JCL to OS/390. Education can take on the form of classroom, self-study, on the job training, on-site, or feet to the fire. The latter being the most undesirable. Consideration should be given to issues such as the availability, cost, length and appropriateness of each. For example, classroom course schedules need to be consulted to determine whether they coincide with project timetables. Cost issues of travel and living expenses need to be also considered. When considering outside consultants for in-house training, they can often tailor classes to specific requirements allowing you to get the most out of this type of education. A list of helpful courses and how to get more course information has been included in Appendix A, “Education Information” on page 535 in this publication.

Highlighted are some of the educational requirements for the key functional areas.

2.6.1.1 System Programming

Education for systems programming personnel will generally include OS/390 installation and tailoring, problem determination and maintenance. Similar education will be required for those with subsystem (for example, CICS or DB/2) responsibility. Another source of education is the hands-on education that occurs when OS/390 is initially installed and before it is put to any kind of productive use. Such hands-on experience has often proven invaluable.

2.6.1.2 Application Programming

Application programming resources will most likely focus on JCL and program development tools for the OS/390 environment. Although there is a high degree of affinity/compatibility between the various programming languages in VSE and OS/390, some education will be needed to understand the functional and compatibility differences that do exist.

2.6.1.3 Operations

OS/390 console operations will be the main education requirement for the operations staff. Courses on OS/390 and JES commands will be the most crucial. Consideration should also be given to education on any scheduling, console automation and/or systems management products that will be used. Operations personnel will also need to be updated on any new procedural and/or process changes.

2.7 Scope of Work and Challenges

When converting VSE applications to OS/390 several tasks have to be performed. The following sections describe the most important work items involved and some of the challenges which can be encountered during the execution of these tasks:

- Application inventory
- Program conversion
- JCL conversion
- File migration
- Automated operations setup
- Project management

2.7.1 Application Inventory

For a VSE to OS/390 conversion, the application inventory is nearly always underestimated in both duration and labor.

The main application inventory activities include:

- Determining what VSE applications must be converted to OS/390
- Retrieving and collecting the current production version of each application item
- Transferring those items to the conversion input libraries on the OS/390 system
- Verifying that the transferred inventory has no missing or unused items

One of the challenges when establishing an application inventory for a VSE to OS/390 conversion is that the application programs must be precisely matched with the JCL streams (for batch applications) and CICS tables (for online applications) to be converted. This is because of the considerable blending of application code and VSE JCL streams (see JCL conversion below). Building these ″work units″ adds work at project start, but it becomes a significant deliverable at project completion, when the new OS/390 application inventory used in production is perfectly defined and centrally stored, with no missing or unused items.

Application inventory tools are used to identify missing and unused items. Missing items must be retrieved or recoded (if possible from a previous version), regression tested under VSE, and used in production under VSE before being converted to OS/390. Unused items must be eliminated or the item using them must be added to the application inventory. The identification, collection and transfer of the application inventory are repeated until the verification identifies no missing or unused items.

The application inventory often leads to a re-organization of the VSE storage. Unique centralized libraries are defined, allocated and used to store the current production version of any application item. Obsolete or duplicate versions are moved to non-production archive libraries.

The application inventory may last two to four months and represent 10 to 15% of the total application conversion effort.

2.7.2 Program Conversion

The conversion of VSE application code to OS/390 is often (but falsely) believed to be the center, most challenging, most labor consuming and most critical part of the conversion, but it is not. With few exceptions (see VSE positioning), it is a simple code modification which does not change program logic, and can nearly always be applied with a simple two-pass translation tool.

VSE COBOL code must also be upgraded to the latest (COBOL for OS/390) compiler level. But this upgrade too, requires no program logic change, and can be applied with a simple two-pass translation tool.

In technical terms, these OS/390 and COBOL upgrade modifications are simple code ″re-engineering″ which fall into one of the following categories:

- Syntax modification: replace a syntax pattern on one or several statements by a similar OS/390 compatible syntax pattern.
- Device independence: eliminate block sizes and other device dependencies from the converted code. Under OS/390, device dependent file attributes are either coded in the JCL or determined by the system managed storage (DFSMS) components of OS/390.
- Elimination of VSE-only features: features such as COMREG, UPSI, DATE and USER can be replaced by calls to user-developed ad-hoc subroutines that simulate the feature under OS/390. Some other VSE-only features, such as the usage of VSE system macros and VSE supervisor calls from Assembler may be more complex to convert.

The difficulty level when converting COBOL code to OS/390 is fairly similar from one VSE installation to another, but it is not so with Assembler. The conversion of Assembler code can be fairly easy, if VSE standard application coding was used, or very complex, if system-dependent non-standard coding was used. In some cases, the conversion of an Assembler program may start with a complete redesign, in which one must identify what function or feature will still be performed by the program, and what function or feature will be handled by the OS/390 system software and utilities. This leads to partial or complete rewrite. Fortunately, those situations are becoming rarer, as VSE installations progressively eliminate their non-standard and system-dependent coding practices.

The conversion of VSE code to OS/390 and COBOL code upgrade may last two to four months and represent 10 to 15% of the total application conversion effort, unless there is a significant inventory of technical non-standard Assembler programming.

2.7.3 JCL Conversion

JCL conversion is nearly always underestimated in both duration and labor. It is the central, most challenging, most labor consuming and most critical part of the VSE to OS/390 conversion.

VSE JCL streams alone are not sufficient to define the flow of the associated job streams. The sequence of steps is evident, but the file references are not always visible. Some are hidden in the standard or partitioned labels. Some are passed and reused from one step to the next. File reference statements (TLBL and DLBL) coded in the JCL are not necessarily used in VSE; the program might not open that file. It is accepted practice in VSE and doesn′t trigger any syntax or execution error. The file open mode (input, output) is not visible from the VSE

JCL: it is hidden inside the code (main or sub-program) associated with the step. Some of the file attributes coded in the VSE JCL are superseded by the disk or tape manager: the proper file attributes must be retrieved in the tape or disk manager′s catalog or in the VTOC listings. In short, it is not possible to understand the flowchart of the job stream without retrieving and analyzing the file opening inside programs and sub-programs, and without collecting in formation from standard labels, partitioned labels, the VSE catalogs and VTOC listings.

Contrary to VSE, OS/390 JCL streams generally reflect exactly the flowchart of the job streams. All files opened within a step have a file reference (DD statement) coded in the JCL. There are no unused file references. The mode of open (input, output, extend) of the file is coded in the OS/390 JCL (disposition).

Therefore, when converting VSE JCL streams to OS/390, whether manually or automatically, ″reverse engineering″ techniques are first used to rebuild the job stream flowcharts from:

- VSE JCL streams
- program conversion (block sizes, device related information, open mode)
- standard and partitioned labels
- VSE catalogs and VTOC listings

This is the most complicated part of the JCL conversion, not only because it requires you to collect and coordinate file reference information coming from different origins, but also because understanding the application job stream requires:

1. Understanding of application data flows (from enterprise-wide cross-references between files and steps)
2. Classification of data flows (that is, files) according to data life cycles:
   - Permanent
   - Handoff
   - Passed temporary file
   - Work (step-level temporary file)
   - Backup
   - External input or output
   - Edition or report
3. Definition and implementation of file management strategies based on the file classification, for example:
   - Usage of GDG for permanent, handoff or backup sequential files
   - Cataloging of passed temporary files and their deletion after last usage
   - Usage of OS/390 ″&&″ work files for step-level temporary files
   - and so on
4. Generation of OS/390 JCL, DFSMS constructs and VSE to OS/390 file migration procedures reflecting the understanding of data flows and their classification.

To illustrate the complexity of VSE JCL conversion and its underlying identification and understanding of data flows, many VSE labels and even physical DASD locations are shared by ″VSE files″ which might (although not always) have the same record length or record layout, but are true separate data

flows. For example: You can have hundreds of files with the same name, for example ″WORK1″. WORK1 can exist in different jobs. Most of the time the WORK1 files will be different from and unrelated to each other. You can however have JobA use a file named WORK1 and JobB running at the same time and also using a file called WORK1. JobA WORK1 gets passed to Job3. Job3 runs and uses WORK1 at a later time. WORK1 in JobB is local use only. The issue is to distinguish the differences between the WORK1 file in JobA and JobC. If these files become intermixed or used by the wrong Job it can be complicated to sort out. There is no data exchanged through those file references. Only analyzing the complete file/step cross-references with associated open modes and attributes allows identifying these situations. When migrating to OS/390, it is critical to identify those data flows as separate OS/390 files. In the worst case scenario, it would create execution or JCL errors. In the best case, it would create unwanted contentions, when concurrent jobs try to access the ″same″ file: OS/390 will allow more job multi-threading than VSE, if contentions are not an issue.

Once the steps above are completed, ″Forward engineering″ techniques are used to generate OS/390 JCL streams that match the application job streams while complying with new OS/390 standards and naming conventions. This is the easiest part of the VSE to OS/390 JCL conversion.

The conversion of VSE JCL to OS/390 may last three to five months and represent 40 to 50% of the total application conversion effort.

2.7.4 File Migration

File migration can only be as good as JCL conversion. This is because the most challenging parts of the file migration, identifying and classifying all files according to their life, and the tape to disk device migration, are for the most part a by-product of JCL conversion.

VSE files and databases can either be migrated ″in-place″ or by copy. Both techniques can be combined in the overall VSE to OS/390 file migration strategy.

In-place migration is by far the quickest, therefore the less disruptive (very short operations outage). Entire VSE data centers with extremely large application data pools have been migrated to OS/390 in an hour. But by altering the VSE production environment, it prevents instant return to VSE. It also complicates, limits or even prevents the implementation of DFSMS, at least at start: natural production cycles may be used to replace the sequential files migrated in place by new DFSMS-controlled versions.

File copy takes longer, but with appropriate configuration and planning, large VSE installations are routinely migrated in mass in only four to eight hours. If the VSE disk space is unaltered by the file migration, instant VSE fallback is possible. This technique facilitates full size DFSMS implementation from the very start of OS/390 operations.

Developing a file migration strategy and associated procedures (VSE and OS/390 file migration JCL streams) is not very difficult, technically speaking. Migrating VSE production files to OS/390 for conversion regression tests allows rehearsing and finalizing the procedures that will later on be used for the actual file migration and operations switchover.

The main challenge is the identification and classification of files for the migration. All files that will be used as input to a job after the switchover to OS/390 operations must be migrated. Files recreated by the first OS/390 production cycles do not need to be migrated, and are better off not being migrated (at least temporary files, cataloged or not).

The task of selecting files for the migration to OS/390 is easier for those files accessed by online applications. This is because they are in relatively small numbers (150 to 300), permanently allocated, often uniquely identified (for example through standard labels), and because their list is fairly stable over time. CICS tables list all those files, and more. The only challenge with online applications is to identify and eliminate obsolete CICS table entries.

The real selection challenge is with batch applications. The list of all files (separate data flows) accessed by batch applications is typically in the hundreds. These files are usually not monitored or kept current. Identification of their use is complicated by reuse of the same VSE file name or even disk space for completely separate data flows. As explained in the JCL conversion section above, it takes a global enterprise-wide view of the step/file cross references to:

- Truly understand the VSE data flows,
- Separate and identify each of them,
- Classify them according to their life cycle (permanent, handoff, backup, work),
- Apply an appropriate OS/390 migration strategy to each one.

Device migration is the second file migration challenge. Many VSE installations tend to be tape (not disk) oriented. OS/390 should be disk and DFSMS oriented, not tape oriented. This means that:

- VSE disk files are migrated to OS/390 disk files
- Most VSE non-backup tape files are migrated to OS/390 disk files, with the exception of external (shipped) input or output tapes
- VSE backup tape files created within application job streams may be migrated to OS/390 tape files. But with DFSMS, they may be created under OS/390 on disk by the OS/390 job stream and copied to tape ″out-of-sync″ with job execution by HSM in a technique called ″disk buffering″ (see OS/390 standards).

It takes a prolonged simulated production test to assess the match of the new OS/390 JCL streams, HSM archival strategies and DFSMS constructs with the available disk space. Hardware configuration constraints and on-going VSE operations do not always allow getting a good feel for the performance of future OS/390 native operations.

The differences in device utilization strategies between VSE and OS/390 greatly influence the file migration. Those differences are defined by the OS/390 standards′ decisions made while converting the JCL.

The VSE to OS/390 file migration is developed progressively over a period of three to five months, while performing the regression tests, and assuming that file identification and device migration are accounted for with JCL conversion, it represents only 5 to 10% of the total application conversion effort.

2.7.5 Project Management

As with application inventory or JCL conversion, the management of a VSE to OS/390 conversion project is nearly always underestimated. The VSE to OS/390 conversion is one of the rare projects that require a coordinated effort from each of the three data processing departments: applications, technical support and operations. When it comes to taking inventory and understanding all the individual items that make up a complete VSE data center, no one has all the answers. Many global answers are obtained by consolidating smaller complementary answers. In fact, in some instances, the participation of the end-users themselves is required.

This is why a VSE to OS/390 conversion must be commissioned, sponsored, and supported by executive management. The Project Manager must be given his overall mission statement directly from the top management, and must be given authority over applications, operations and technical support for this project.

One of the challenges of managing a VSE to OS/390 conversion is project planning. The conversion of VSE applications (JCL, programs and files), associated testing and implementation (switchover) are complex in themselves. It may involve 10 to 20 people. The project plan averages 150 tasks and sub-tasks, most of them linked through dependencies. It becomes even more complex, when this plan must be coordinated with the detailed OS/390 software installation and implementation plan, the staff education plan, the OEM (non IBM) software installation and implementation plan, and the parallel application maintenance and development plan. The data center doesn′t come to a stand still while the VSE to OS/390 migration takes place.

Finally, resource management, both human and configuration-wise, can be a real challenge. Hiring conversion experts to handle parts of this one-time project can be part of the solution for human resources constraints. The project still requires a significant internal human resource investment to handle a number of activities that are best left to the data center personnel itself. This is true for application inventory (sorting out duplicate program versions and so on), OS/390 standards decision that define the key operating processes (naming conventions, device migration, and so on), and regression testing (test plan and scripts and so on).

Project management represents 10 to 15% of the total application conversion effort.

2.7.6 Automated Operations

In recent years, the setup (population) and implementation of a job scheduler and report manager have become a full part of the VSE to OS/390 migration. Regardless of your VSE implementation of a job scheduler and report manager, in OS/390 they will be used for the entire production, all jobs, all reports.

Identifying and carrying over the report management instructions from the VSE JCL (destination, number of copies, FCB, and so on) to the OS/390 report manager is not very challenging. Neither is carrying over existing job scheduling or report management instructions from a VSE to an OS/390 product.

The real challenge is to learn not only how the OS/390 product works, but also how to use it. The OS/390 basic education provided by vendors of OS/390 job schedulers and report managers is just that: ″basic″. Even with hands-on exercises, it doesn′t prepare the production control staff who attend it to design and define on their own how they will use the product to implement operation

procedures. Most will simply try to reproduce with the new OS/390 product what they were doing in VSE with or without assistance of a product. The challenge is to:

- Understand how OS/390 works,
- Understand how the OS/390 job scheduler or report manager is best used,
- Define specific local implementation rules and guidelines (standards), and finally
- Convert the existing VSE instructions and ways of doing things from VSE to OS/390.

An additional complication is that it is difficult to test the population of those products (at least for the report manager) in simulated production mode without disrupting or confusing the end-users. Test versions of application reports created under OS/390 cannot be sent to their future recipients using the OS/390 report manager without risking that they be taken as current VSE generated production versions. It is feasible to verify most of the automated daily job scheduling by simply running the OS/390 job scheduler in simulated production mode, although it is never easy to reproduce all actual production size executions and event triggered executions. But it becomes a real challenge to mimic lower frequencies such as weekly, biweekly, and monthly, especially when they are integrated to daily production. In any case, if those products are to be used in production under OS/390, they must be used when regression testing the converted applications.

It is atypical that the OS/390 job scheduler and report manager will be:

- Populated once, just after the vendor′s basic education class
- Changed partly or totally a few months later, after regression testing has identified a number of conceptual or implementation flaws
- Adjusted one more time after switchover, once in OS/390.

This is because production control personnel are often ill prepared to perform this migration. Participation of hired consultants, experts with the OS/390 product implementation or an application analyst or technical support staff may be part of the solution.

The setup of a job scheduler and report manager may last three to four months and represent 10 to 15% of the total application conversion effort.

2.8 Cost Considerations

It is often thought that OS/390 will require more hardware and staff resources. While OS/390 may, in some cases, require more overhead and hence CPU, per unit of work, typically greater system throughput is achieved over that of the VSE environment. Due to enhanced systems management and automation capabilities it has been found that OS/390 staff requirements actually do not increase compared to VSE. In cases where staff increases have been seen, it has usually resulted from growth in system requirements. That is, application and end-user growth requirements has spawned the need for additional system resources. This need for additional system resources, then, sometimes requires additional human resource support.

While migration project cost projections will vary for each environment and customer, there are some basic cost elements that are common to all projects.

The purpose here is not to predict or estimate project costs but to identify major cost elements and any relevant financial resource considerations.

- Cost/Benefit Requirements
  - ► Reasonable and predictable timeframe
  - ► Reduced internal staff participation focused on learning OS/390
  - ► No delay/postponement of development and maintenance
  - ► Controlled costs turned into investment
  - ► Low risk
- Migration project cost elements
  - ⇒ General
    - Education
      - Course fees
      - Travel & living expenses
    - Consultants
    - Internal human resources (chargebacks)
      - Project manager
      - Team members
  - ⇒ Hardware
    - Incremental/interim configuration to support migration
      - LPAR (CTCs, channels, device channel adapters, EMIF)
      - Separate footprint (w/ additional software licenses)
    - Final configuration
  - ⇒ Software
    - Incremental/interim configuration to support migration
      - VM
      - Conversion Tools
      - VSE & OS/390 licenses
    - Final OS/390 configuration (including optional products & ISV products)

2.9 OS/390 Documentation Resources

OS/390 documentation resources should be consulted as early on in the project as possible. This should be done in order to get an understanding of some of the issues associated with installing and implementing the OS/390 environment. For example, it will be necessary to understand the various OS/390 delivery mechanisms (that is, CBIPO, ServerPAC, SystemPac) in order to determine the one most appropriate based upon the given requirements/environment.

2.9.1 Introduction References

- Key CD-ROM Collections (Bookshelves) for OS/390
- General Information Manual (Introduction and Release Guide)

2.9.2 Key Documents and Other References

- *OS/390 V2R5 Planning and Installation*, SK2T-2484
- CBIPO (System Pak) Custom Built Offerings Planning
- CICS Up and Running
- DB2 Release Guide

2.9.3 Web URL

http://WWW.S390.IBM.COM/OS390/

Chapter 3. Developing the Plan

This chapter discusses the following topics:

3.1 Overview

3.1.1 References

These materials provide sources of supplemental information for this chapter.

- *MVS Migration System - Planning Guide*, SB11-8077 describes the planning process for the MVS-MS. This guide is for the people who are responsible for planning and scheduling the migration and fitting the conversion that MVS-MS performs into the migration schedule. It is the basic book for the project manager and every technical person involved in planning and running both the migration and the conversion.
- *MVS Migration System - General Information*, GB11-8074 provides an overview of the IBM MVS Migration System and is for the people at an installation who will decide if MVS-MS will work for a particular environment. It describes both the advantages and limitations of MVS-MS, presents information on how MVS-MS works, and identifies some specific early planning concerns.
- *MVS Migration System - Planning Chart*, SB11-8090 displays the standard conversion tasks and subtasks relative to their duration and relationship to each other.

3.1.2 Recommendations

The following are recommendations for your migration that are not project phase specific or apply to all phases of migration.

3.1.2.1 Project Management

In some cases it may make sense to hire contractors, temporary personnel or a service provider to perform tasks that will only be performed once and do not provide long term payback to the installation. These one time tasks may include project management, specific conversion activities and use of project specific tools. There are many tasks to consider during a migration. Careful consideration should be given to knowing the skills that are available to the project, the requirements for systems programming, other projects that are planned or in progress, and how augmenting these skills and personnel may or may not make sense.

3.1.2.2 Take Advantage Of Conversion Tools and Automation

Executing a migration with a mass conversion tool and automated processes can reduce both the time and people required to migrate from VSE to OS/390. Where it is not a large task to convert three programs and two strings of JCL, it is a large and difficult task to increase the scope by one thousand and perform the same conversion.

The automation provided by the use of a mass conversion tool is unique. After an extensive period of analysis, which includes running both pilot conversions and dummy conversions, you can, in a final mass conversion, convert all of your VSE applications to MVS in a single automated process.

3.1.2.3 Migration Plan - Guide and Outline

Creating a migration plan involves analyzing what a migration requires and developing a plan to customize the general process to your particular installation. Developing a comprehensive and detailed migration plan is important to the success of a migration.

The type of conversion method directly affects the content of your plan. For this guide we have chosen to follow a mass conversion method using the Cortex MS processes. Chapter 3, Developing the Conversion Plan, of the *MVS-MS Planning Guide* provides information on how to develop a migration plan where a mass conversion method is used. Use it as a guide to develop a plan that is specific to your site.

Appendix A of the *MVS MS Planning Guide* provides the outline of a sample conversion workbook that you can use to write your conversion plan. The model workbook contains a checklist and some questions to help you generate ideas on what to include in your conversion plan.

3.4, “Plan Examples” on page 53 also provides a sample conversion project plan.

3.1.2.4 Two Phase Approach

The migration project can be broken into a few logical pieces that may help its execution. One method that has been successful is to begin with a mini project, phase 1, to identify and resolve your inventory. Proceeding with a known inventory will allow more precise pricing. The pricing for a conversion effort is based on inventory. It also provides information about the effort that may be required to recreate source materials. There are tools and service providers that perform these services. The second phase is the actual implementation.

The Phase 1 output is also a standalone deliverable that can be very useful for Year 2000 preparation.

3.1.2.5 Conversion Method

There are two basic approaches to the migration. One approach, referred to as the kernel approach, converts a single application or subsystem at a time. The other, called mass migration, converts all applications, the entire system, at the same time. The method or approach used will dictate the elements of the project plan. This chapter will explore the major considerations of using mass migration as a conversion method and as a conversion tool.

Two tools support or implement the mass migration approach. One of these tools, the IBM MVS-Migration System (MVS-MS), was previously licensed from

SISRO and is no longer available, but deserves mentioning. The product documentation is helpful in that it provides a very good project plan and description of the mass migration approach. When sold through SISRO, this tool is know as CORTEX-Migration System (CORTEX-MS) and currently is available. Although there have been many changes to the MVS and VSE operating systems and improvements to the conversion tool, the methodology of planning and execution of the conversion has not changed significantly.

Choosing the appropriate conversion and production implementation strategy is a very crucial decision. It is important to choose the right strategy and build a corresponding plan. The mass migration method can provide a project that is definable and allows for more accurate project cost estimation and sizing. It can be the most effective strategy in light of today′s I/T structure where integrated applications are closely tied to the integrated functions of business operations.

3.1.2.6 Project Staffing

It is recommended to use hired conversion specialists to handle the planning and organization of the overall OS/390 migration, and the conversion of the VSE applications to OS/390.

The VSE staff and hired conversion specialists work as a single project team. Each bring their own skills to the project and share the project responsibilities as follows:

3.1.2.7 Librarian

The librarian helps the project manager follow the migration by recording events, collecting information about the progress of the migration, drawing up checklists, and maintaining tables of problems, solutions, and programming elements affected. The tasks and responsibilities of the librarian include:

- Controlling the production and updating of the migration workbook
- Collecting information on the VSE source material
- Recording migration events
- Collecting information on program and JCL conversion, and conversion problems and solutions

3.1.2.8 Migration Responsibilities

The ***hired conversion specialists*** are typically skilled and experienced with:

- OS/390 applications and operations support
- OS/390 installation and implementation
- VSE to OS/390 conversion tools
- VSE to OS/390 conversion requirements and solutions
- OS/390 migration planning and project management

The ***VSE staff*** is experienced with:

- Existing VSE operations
- Existing VSE applications
- Existing OS/390 applications (in case of pre-existing dual VSE and OS/390 operations)

The ***hired conversion specialists*** can be deployed for converting the in-house developed applications, and leading the overall migration effort, including:

- Following the migration methodology and the project plan
- Identifying and addressing the conversion requirements
- Converting the VSE applications to OS/390
- Design and delivery of a state-of-the-art standardized OS/390 environment

The ***VSE staff*** is mainly responsible for:

- The new OS/390 HW/SW configuration
- The VSE application inventory
- Regression testing the converted applications
- OS/390 migration activity outside the conversion of VSE application code, JCL or files

3.1.2.9 Migration Assignments

The ***hired conversion specialists*** are typically assigned the following application conversion tasks:

- Manage the overall migration project
- Manage their own team and responsibilities
- Provide technical leadership, including project planning
- Receive and validate the application inventory
- Develop the conversion specifications
- Customize the conversion tools to local requirements
- Develop new conversion tools (if applicable)
- Perform manual conversion activities, when automation is not possible or not cost efficient
- Perform automated mass conversions
- Assist with setup of OS/390 automated operations tools
- Participate in online applications tests
- Participate in batch applications tests
- Participate in applications switchover to OS/390
- Support initial OS/390 operations

The ***VSE staff*** can be assigned the following application conversion tasks:

- Manage their own team and responsibilities
- Provide office space and project support tools
- Participate in project planning
- Receive OS/390 basic education
- Provide and install OS/390 HW/SW resources
- Operate the OS/390 environment
- Design and implement security
- Migrate the CICS application tables, and the network
- Collect and supply the application inventory
- Assist with the conversion specifications
- Participate in VSE positioning activities, when automation is not possible or not cost efficient
- Install, setup and operate OS/390 automated operations tools
- Provide, install and test the OS/390 version of purchased applications
- Modify all interfacing systems (PC-LANs, RJE, NJE, ...) to reflect the OS/390 migration
- Perform online applications tests
- Perform batch applications tests
- Participate in applications switchover to OS/390
- Run and support initial OS/390 operations

3.2 Plan Components

3.2.1 Approach

For the purposes of providing more specific guidance for conversion projects, an approach to the migration had to be determined. This is also true for the migration effort itself, an approach must be adopted. In these discussions, we will describe the environment associated with using the Mass Conversion methods and tools.

3.2.2 Team

Before the actual project plan is developed, thought needs to be given to the project/migration team and the functions, responsibilities and composition of this team. There are many different ways to organize a migration team, the group of people responsible for planning and executing the migration project. A recommended organization for the migration team (see Figure 5) consists of the following people:

- A project manager, who is responsible for the migration procedure as a whole - general specifications, planning, coordination, and follow-up.
- Two systems or applications programmers, or one from each area, who draw up detailed migration specifications, install and customize any mass migration/conversion tools.
- Two operations people to take charge of conversion testing.
- A librarian to help control and track the migration activity.

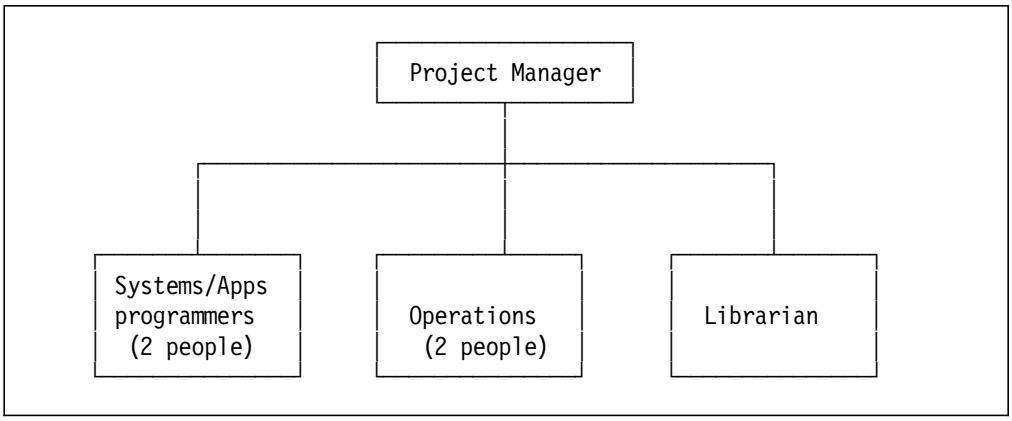


Figure 5. Migration Team

The migration team should include people with the following knowledge or skills:

- Some knowledge of the concepts and facilities of the OS/390 system
- Knowledge of the current VSE environment and the applications to be converted
- Development or systems skills to analyze special situations encountered during the early phases of the migration
- Operations skills to test converted applications under OS/390 and to assess the impact of converted operational procedures on the OS/390 productions operations environment

If a mass migration/conversion tool is used someone will need to become familiar with the product and the mass migration method. The team members should consult the product documentation related to their responsibilities and run the sample conversion.

The functions and responsibilities of each member of the team depend to some extent on local conditions. The following sections describe, in general terms, the tasks each member may perform.

3.2.2.1 Project Manager

The project manager manages the project as a whole, selecting the key resources, both technical and non-technical, required for the project. The project manager should posses the appropriate project management skills and should have current knowledge of project management techniques and tools. The project manager could be a systems programmer or technical manager who is knowledgeable in:

1. OS/390
2. The mass migration tool
3. The applications to be converted
4. VSE

The project manager's tasks and responsibilities include:

- Managing and controlling the migration project
- Acting as a liaison between the migration team and others within the I/S organization
- Drawing up migration specifications
- Designing migration procedures
- Tracking migration schedules and assigning necessary resources
- Determining any conversion tool customization
- Planning and preparing for the OS/390 production switchover

3.2.2.2 Systems Programmers

The systems programmers help the project manager to design migration procedures. They must resolve technical problems related to the local implementation of both VSE and OS/390; therefore, they must be familiar with both VSE and OS/390. Their tasks and responsibilities include:

- Helping to design the specifications for the migration
- Helping the project manager design the migration procedures
- Installing and customizing any conversions tools
- Analyzing and solving conversion problems
- Preparing for the OS/390 production switchover
- Assisting with OS/390 operations

3.2.2.3 Applications Programmers

The applications programmers help the project manager to develop migrations procedures. They also test converted applications. They should be thoroughly familiar with critical applications (both online and batch) and understand both VSE and OS/390. Their tasks and responsibilities include:

- Helping to design the specifications for the migration
- Analyzing and preparing the VSE source material
- Developing any conversion tools or specific conversion procedures
- Manually converting some general purpose user routines and programs
- Analyzing and solving conversion problems

3.2.2.4 Operations

If a mass migration tool is used, operations personnel will submit and control the mass migration jobs, complete and check the production database, and test the converted applications under OS/390. They must understand the operating procedures of VSE and OS/390, and know how to use the tool. Their tasks and responsibilities include:

- Helping to design the specifications for the migration
- Designing jobstep preparation
- Preparing VSE files and JCL
- Implementing conversion and OS/390 operational procedures
- Testing the converted applications
- Completing and checking the mass migration tool output
- Assisting with OS/390 operations

3.2.3 Tasks

It cannot be stressed enough how absolutely important a well thought out and well documented project plan is to the successful completion of the migration project. Discussed here will be some of the key essentials in planning for such a project and some thoughts on how the actual project plan should be developed. Assistance with developing the conversion plan can be found in Chapter 3 of the *MVS MS Planning Guide* named ″Developing the Conversion Plan″. The checklist that was used to develop that plan can also be found in Appendix A , ″The Conversion Workbook″ of that publication. An example of a project plan can be found in 3.4.2, “Project Plan Example” on page 56.

Listed below are some of the main tasks that are involved in a migration.

1. Defining objectives.
2. Analyzing what the tasks required in a migration are and developing a **well-documented migration plan**.
3. Assigning personnel to the conversion team.
4. Deciding on a conversion method and a conversion tool(s).
5. Analyzing the VSE workload and developing a comprehensive list of the applications to be converted.
6. Planning for and upgrading hardware.

7. Training personnel to work with the OS/390 system.
8. Planning and installing the OS/390 system products.
9. Developing standards for application conversion that reflect such things as naming conventions to be used in the new OS/390 system environment.
10. Collecting all VSE source materials and presenting same to the conversion process.
11. Translating VSE programs to OS/390 programs.
12. Converting JCL.
13. Transferring data files from VSE to OS/390.
14. Testing each converted application under OS/390.
15. Documenting and preparing run books and operational procedures.
16. Implementing the production workload under OS/390.

3.2.4 Milestone Events

Within each migration, certain activities should be identified as key activities, the attainment of which can signal significant progress (or the lack of attainment, a schedule slippage). These activities are typically called milestone events or just milestones. Each customer should identify the milestones most important in their migration.

The following are some suggested VSE to OS/390 migration milestones:

- Migration Plan completed, reviewed, and approved.
- VSE software inventory completed.
- All vendor support committed.
- Essential education completed.
- Necessary hardware installed.
- Installation of OS/390 and related products completed.
- Initial OS/390 IPL performed.
- Pilot conversion completed.
- Each major application's successful conversion.
- Stress/Production tests completed.
- Operator education and Run Books completed.
- Production criteria attained.
- Production implementation initiated.

Once milestones are defined, periodic "checkpoints" can be scheduled to monitor successful milestone completion; that is, project progress.

3.2.5 Education

OS/390, and to some degree migration/conversion skills are crucial factors to the success of the migration project. Identification of skill requirements and how these requirements will be satisfied is the main objective of the education plan. Listed are the key elements to effective education planning:

- Identify personnel (who)
- Identify personal needs (what)
- Set schedules (when)
- Map a master plan (how)
- Identify resources/offering dates

3.3 Progressive versus Mass Conversion

3.3.1 Approach Differences

The difference between the progressive and mass conversion approaches is illustrated on Figure 6.

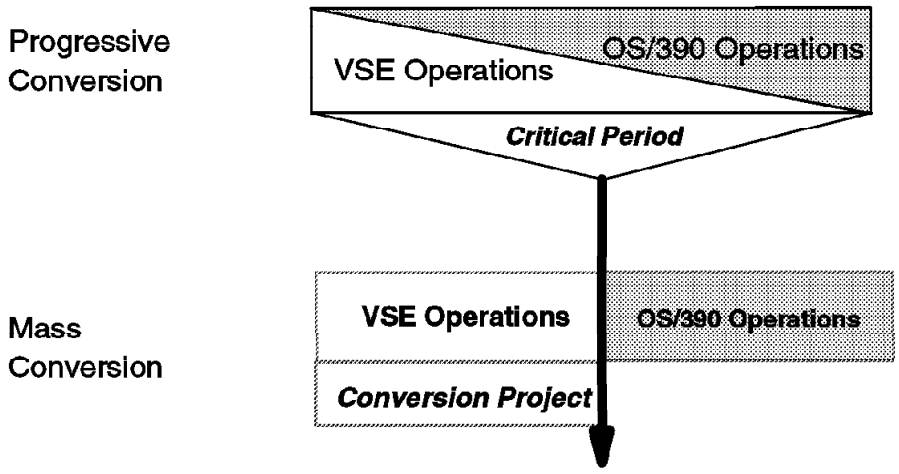


Figure 6. Progressive versus Mass Conversion

In a progressive conversion, the VSE application portfolio is divided in smaller application units (or kernels) which are migrated one by one to the target OS/390 environment. The production is divided between VSE and OS/390 operations for an extended period of time. During that critical period, the OS/390 system supports simultaneously the new production and the application conversion including the conversion testing.

The main distinction with the mass conversion approach is that it results in a single switchover of the entire VSE application portfolio to OS/390 over a weekend, with no overlap of VSE and OS/390 production. Until the switchover weekend, all converted applications run in production under VSE. By the end of the switchover weekend, all converted applications run in production under OS/390. An OS/390 system is used in parallel with ongoing VSE operations to support the migration project, but it doesn′t support any OS/390 production until the switchover weekend.

3.3.2 Historical Perspective

The progressive conversion approach was the only solution available until the early 80s.

More recently modern VSE operations have substantially grown in size, complexity and integration, making it more difficult to implement a progressive conversion.

It is also because the mass conversion approach, which was in a pioneer stage in the early 80s, has matured to become safe and proven alternative. Hundreds of mass conversions have been successfully completed worldwide in the past 15 years.

3.3.3 Shared Application Files and Databases

With today's highly integrated VSE application portfolios, it becomes increasingly difficult to define isolated application kernels for a progressive conversion. Most applications share access to the same permanent files or databases. If some files and databases need to be accessed at the same time by some application kernels running in production under VSE and other application kernels running in production under OS/390, those files and databases must be duplicated under VSE and OS/390. The duplicate versions must then be kept in sync, which requires developing complicated application bridges between VSE and OS/390. The bridges must constantly be changed, as application kernels are progressively migrated from VSE to OS/390.

3.3.4 Shared Application Code

A similar challenge exists for reusable code, such as JCL procedures, subroutines, macros, copybooks and includes. Duplicate versions must be maintained under VSE and OS/390 while application kernels sharing usage of those code items run on different operating systems. Duplicate source storage systems and change control procedures must be maintained during the overlap.

3.3.5 Operations Support Staffing

Supporting and operating dual VSE and OS/390 production environments requires a larger staff and skill set than for a single production environment.

3.3.6 Automated Operations Tools

The complexity and sophistication of modern VSE operations shows in the catalog of automated operations tools on which they rely. Those tools often include a job scheduler, a report manager and a tape manager, which complicates the organization and implementation of a progressive conversion.

It is very challenging to coordinate the overall job scheduling when two synchronized and inter-dependent parts of the application portfolio run on two separate operating systems under the automated control of two separate job schedulers. Job schedulers are not designed or able to coordinate production between two separate operating systems. In addition, as discussed above for shared permanent data file and databases, the on-going progressive migration of application kernels, forces to constantly change the automated job scheduling on each side.

A similar challenge awaits progressive conversion teams with the division of report management instructions between two report managers running on two

separate operating systems, or the division of tape files and tape volumes between two tape managers running on two separate operating systems.

3.3.7 Standardized Conversion Deliverables and Automation

A significant objective for today′s VSE or OS/390 mainframe installation is the standardization of their application components (JCL streams, application code and data files), associated naming conventions and operation procedures. The standardization of conversion deliverables is directly related to the degree of automation used to perform the conversion. The more automation is used, the more standardized the deliverables will be. Mass conversions are typically more automated than progressive conversions.

It is also much easier to guarantee complete and consistent compliance with standards and naming conventions when the entire inventory is converted and switched from VSE to OS/390 over a single weekend using a single automated conversion process, as in the mass conversion approach. Contrarily, it is difficult to guarantee a good compliance with standards and naming conventions when the conversion of application kernels spans over many months and may be assigned to separate conversion teams, as in a progressive conversion. The same conversion requirement may be addressed differently by different people at different times.

3.3.8 Risk Management

The comparative risk of both conversion approaches has changed over the years.

The risk of disrupting your production system, when dividing it into dual operating environments, has increased in proportion with the VSE application portfolios increase in size, complexity and integration.

With mass conversions, the regimen of performing multiple successful rehearsal conversions has refined the mass conversion approach and its single switchover weekend into a mature and predictable, therefore safer solution.

It is today safer to use the mass conversion approach than the progressive one for large application portfolio, and in some cases of high integration, there is simply no other way.

3.3.9 Complexity of Implementation

Still, the mass conversion approach requires more skills and experience than the progressive conversion approach.

The conversion of one single application kernel requires less integrated automation, therefore less complex (and less expensive) conversion tools. Due to the reduced size of a kernel, it is fairly easy to recover manually from automated conversion defects. The migration of a single kernel requires less planning than the conversion of the entire portfolio. Consequently, the progressive conversion approach has an easier learning curve, which makes it easier to implement with internal non-conversion-expert staff only. They learn while they do it.

Contrarily, the mass conversion approach requires highly integrated automation, therefore complex and expensive conversion tools. Due to the size of the conversion inventory, it is difficult or impossible to recover manually from

automated conversion defects. The switchover of an entire VSE production to OS/390 over a weekend cannot be improvised: it requires perfect precision and planning by experienced conversion specialists. The complexity of powerful mass conversion tools makes it difficult to staff with internal non-conversion-expert staff exclusively, because of the learning curve. Hiring conversion consultants experienced with mass conversions and single weekend switchovers is highly recommended for users who want to migrate large integrated VSE production environments to OS/390 over a single weekend.

3.3.9.1 Mass Migration as a Conversion Method

Mass migration uses the single switchover method of migrating a VSE installation to OS/390. The various conversion tasks that need to be performed using this methodology are described in the MVS-MS and CORTEX-MS documentation. The conversion method, or process, consists of running three conversions:

1. The pilot conversion - a conversion of a small subset of the VSE applications, usually involving all or part of the most important work. The pilot conversion educates project team members, provides the time to define OS/390 standards, code any exits deemed necessary for customizing, and overall prepares the team for the rest of the conversions.
2. The dummy conversion - a conversion of all VSE applications; a process that is normally repeated many times as changes are made to VSE source materials over the life of the project. This is why ″freezing″ VSE application maintenance is not necessary with this methodology.
3. The actual mass conversion (or switchover) - a conversion of all VSE applications, the switchover of the VSE files and catalogs, followed by OS/390 production operations.

Because MVS-MS and CORTEX-MS documentation guides you in the steps and tasks to be performed, it helps you develop a comprehensive and detailed migration plan for your own installation. The documentation also provides the skeleton migration plan with staffing recommendations - you provide your installation-specific details.

3.3.9.2 Mass Migration Used as a Conversion Tool

Used interactively, MVS-MS or CORTEX-MS is a set of subsystems that are panel driven and use TSO terminals to direct all conversion tasks. The tool translates VSE source programs written in the following languages:

- Assembler
- COBOL
- PL/I Optimizer
- RPG II

In addition to translating the above VSE programming language programs into OS/390 source equivalents, MVS-MS or CORTEX-MS, herein after referred to as the tool, also performs the following:

- ISAM programs are translated to VSAM; that is, ISAM I/O statements translated into VSAM statements.
- COMREG, CNTRL, and PRTOV functions (VSE functions not directly supported in OS/390) are simulated under OS/390 by the tool’s simulation routines.

- Program clauses that restrict device independence are eliminated; that is, I/O assignment clauses removed from programs, placed in JCL.
- Program console interactions (for example, COBOL DISPLAY/ACCEPTs) are removed from being executed at program runtime; rather this input is requested at job setup time via job preparation panels and prompts.

The tool converts VSE JCL (procedures, standard label definitions) and the POWER Job Entry Control Language (JECL) - $$LST, $$PRT, $$PUN, $SLI - into OS/390 JCL jobstreams. VSE uses of standard utilities are translated into OS/390 equivalents - SORT/MERGE, IDCAMS, and IEBGENER utilities.

The tool lists information in cross reference reports that enables the installation to make sure that the VSE input libraries are complete. The information provided includes lists of relationships:

- Between JCL and PROC/SLI books
- Between JCL (or PROC/SLI books) and programs, PSB, DBD, or FCB definitions
- Between programs and called modules
- Between programs and copied members or macros

The above data can help the project team determine ″what′s missing,″ ″what′s duplicated,″ and ″what′s not used″ of the VSE source materials. (It can′t help the team find missing source, however.) Thus, the tool can assist in one of the most crucial tasks of the migration; that is, reconciling the ″source VSE materials″ needed for the conversion process. This is sometimes referred to as the Data Analysis Phase.

In addition to source program and JCL translation, the tool also provides:

- OS/390 standards naming convention assistance
- Testing facilities to help when testing converted programs
- Operator job preparation and submission panels
- Exits for the tool customizing purposes
- Operator job logging facilities
- Online terminal exercises to help in learning the tool operations

3.4 Plan Examples

The following is a sample plan for the migration of ABC Company from VSE to OS/390. ABC Company will be contracting OS/390 services from SER company. CNV Company has been contracted to provide professional migration services and will be using CORTEX-MS.

3.4.1 Project Schedule

3.4.1.1 Estimated Project Schedule

The following is an estimated schedule for Project 2 - VSE to MVS conversion. The project may begin upon completion of the Inventory Determination task of Project 1, estimated to be on or about June 1, 1996, and will last approximately nine (9) months with a switchover to MVS after approximately eight (8) months.

Table 3. Nine Month Project

Month Number	1	2	3	4	5	6	7	8	9
Month Initial	**J**	**J**	**A**	**S**	**O**	**N**	**D**	**J**	**F**
Phase 1 - Specifications	**	**	**	*					
Phase 2 - Custom Modifications of CORTEX-MS	*	**	**	**					
Phase 3 - First Trial Conversions: Online and Batch Appl			*	*					
Phase 4a - MVS Tests & Repetitive Conversions : Online				*	**	**	**		
Phase 4b - MVS Tests & Repetitive Conversions : Batch				*	**	**	**	*	
Phase 5 - Actual Conversion and Switchover								*	
Phase 6 - Assisted MVS Operations									**

3.4.1.2 Estimated Schedule for CNV Responsibilities

The following is an estimated schedule for CNV responsibilities. The actual schedule will be determined at project start.

Table 4. CNV Responsibilities

Month Number	1	2	3	4	5	6	7	8	9
Month Initial	**J**	**J**	**A**	**S**	**O**	**N**	**D**	**J**	**F**
01 - Manage CNV Conversion Responsibilities	**	**	**	**	**	**	**	**	**
02 - Provide Technical Leadership	*		*	*				*	*
03 - Receive and Validate the Conversion Inventory	*	*	*	*	*	*	*	*	*
04 - Develop the Conversion Specifications	**	**	**	**	*				
05 - Custom Modify CORTEX-MS and Proprietary Tools	*	**	**	**	**				
06 - Perform Manual Conversion Activities		*	*	*					
07 - Perform Automated Mass Conversions	*	*	*	*	*	*	*	*	
08 - Assist with Setup of MVS Automated Operations Tools				*		*		*	
09 - Participate in Online Applications Testing				**	**	**	*		
10 - Perform Batch Applications Testing				*	**	**	**	*	
11 - Participate in Applications Switchover to MVS							*	**	
12 - Support Initial MVS Operations									**

3.4.1.3 Estimated Schedule for ABC Responsibilities

The following is an estimated schedule for the ABC responsibilities. The actual schedule will be determined at project start.

Table 5. ABC Responsibilities

Month Number / Month Initial	1 J	2 J	3 A	4 S	5 O	6 N	7 D	8 J	9 F
01 - Manage ABC Conversion Responsibilities	**	**	**	**	**	**	**	**	**
02 - Participate in Project Planning	*	*	*		*		*		
03 - Operate MVS Environment	**	**	**	**	**	**	**	**	**
04 - Provide Office Space and Project Support Tools	*								
05 - Determine and Supply VSE Material to be Converted	*	*	*	*	*	*	*	*	
06 - Assist with Conversion Specifications	**	**	**	**					
07 - Apply Manual Modifications to VSE Material (If any)		*	*	*					
08 - Set up MVS Operations Tools				**	**	**	**	**	
09 - Perform Online Application Tests				**	**	**	**		
10 - Perform Batch Application Tests				*	**	**	**	*	
11 - Participate in Applications Switchover to MVS								**	*
12 - Support Initial MVS Operations									**

3.4.1.4 Estimated Schedule for SER Responsibilities

The following is an estimated schedule for the SER responsibilities. The actual schedule will be determined at project start.

Table 6. SER Responsibilities

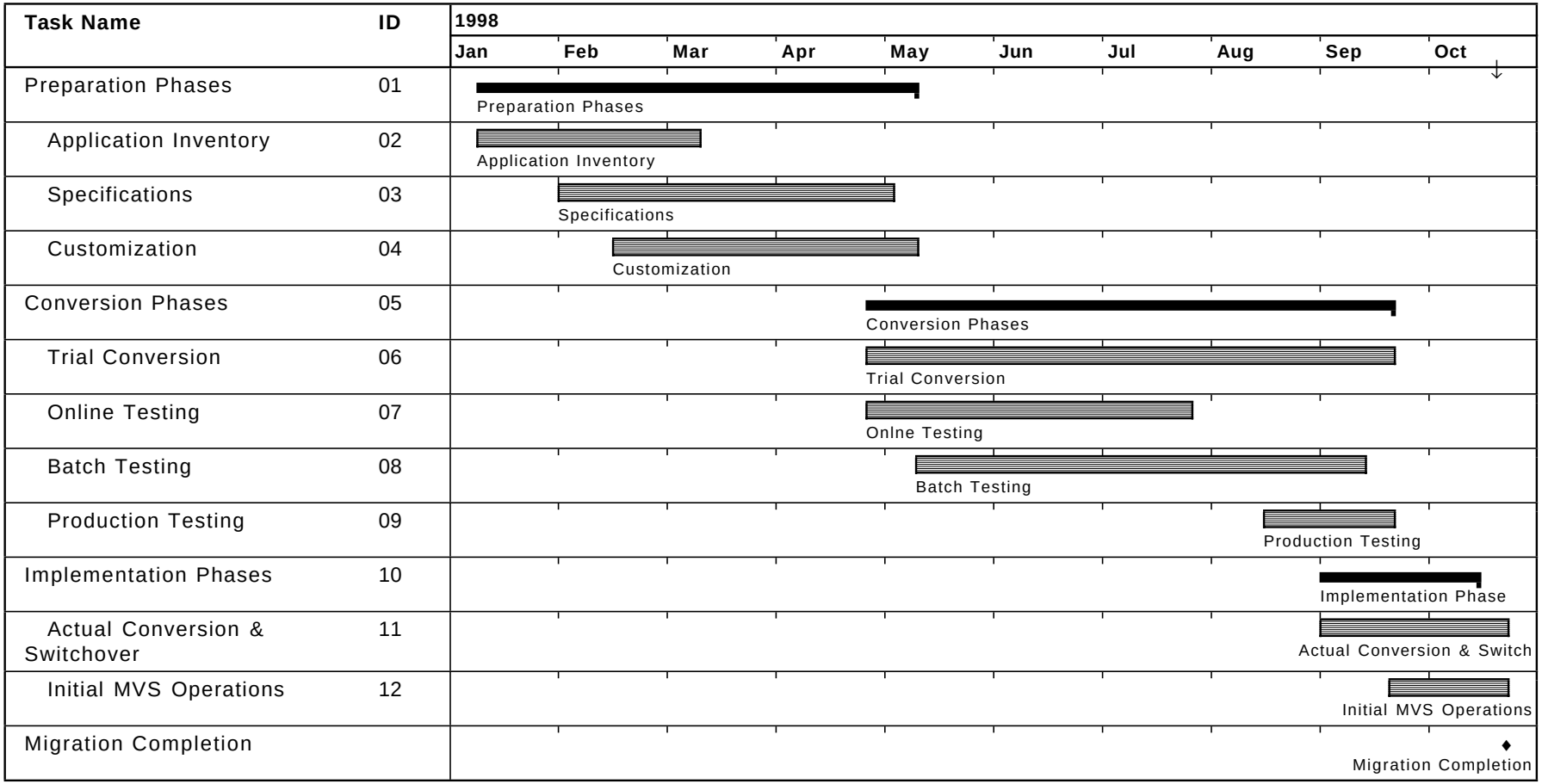
Month Number / Month Initial	1 J	2 J	3 A	4 S	5 O	6 N	7 D	8 J	9 F
01 - Provide MVS Resources	*		*	*			*		
02 - Install, Maintain, and Support MVS Environment	**	**	**	**	**	**	**	**	**
03 - Assist with Conversion Specifications	**	**	**	**					
04 - Participate in Applications Switchover to MVS								**	*
05 - Support Initial MVS Operations									**

3.4.2 Project Plan Example

The actual schedule will be determined at Project 2 start, based on the completion of the Project 1 Inventory Determination completion date, the expected switchover date, and any potential conflicts with ABC′s processing.

3.4.2.1 Project Plan - Summary

Task Name	ID	Projected		Actual	
		Start	End	Start	End
Preparation Phases	01			01/09/98	05/10/98
Application Inventory	02	01/09/98	03/10/98		
Specifications	03	02/01/98	05/03/98		
Customization	04	02/15/98	05/10/98		
Conversion Phases	05			04/26/98	09/21/98
Trial Conversion	06	04/26/98	09/21/98		
Online Testing	07	04/26/98	07/26/98		
Batch Testing	08	05/10/98	09/13/98		
Production Testing	09	08/16/98	09/21/98		
Implementation Phases	10			09/01/98	10/15/98
Actual Conversion & Switchover	11	09/01/98	10/23/98		
Initial MVS Operations	12	09/20/98	10/23/98		
Migration Completion				10/23/98	10/23/98

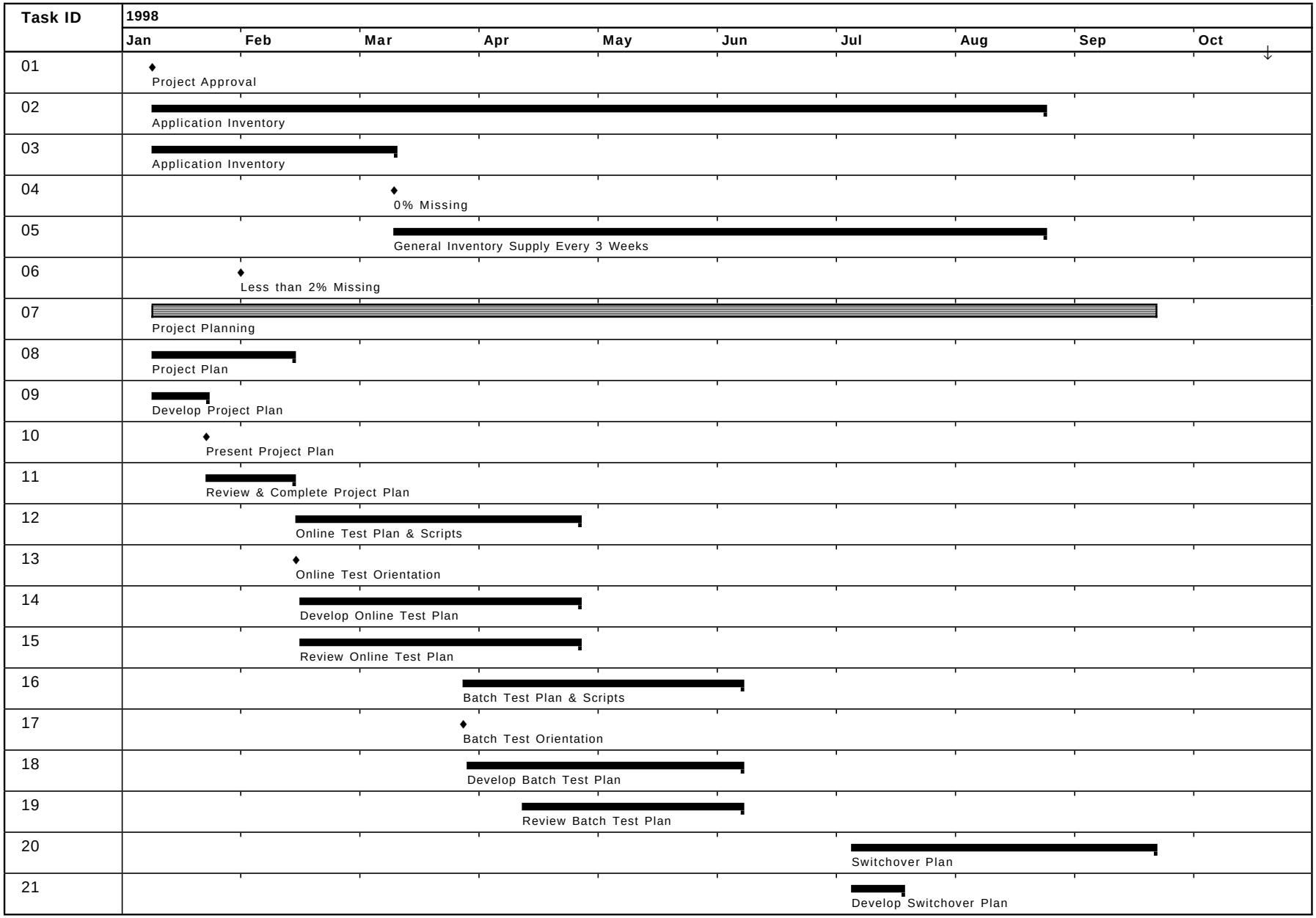

Task Name
ID
1998
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Preparation Phases
01
Preparation Phases
Application Inventory
02
Application Inventory
Specifications
03
Specifications
Customization
04
Customization
Conversion Phases
05
Conversion Phases
Trial Conversion
06
Trial Conversion
Online Testing
07
Onlne Testing
Batch Testing
08
Batch Testing
Production Testing
09
Production Testing
Implementation Phases
10
Implementation Phase
Actual Conversion & Switchover
11
Actual Conversion & Switch
Initial MVS Operations
12
Initial MVS Operations
Migration Completion
Migration Completion

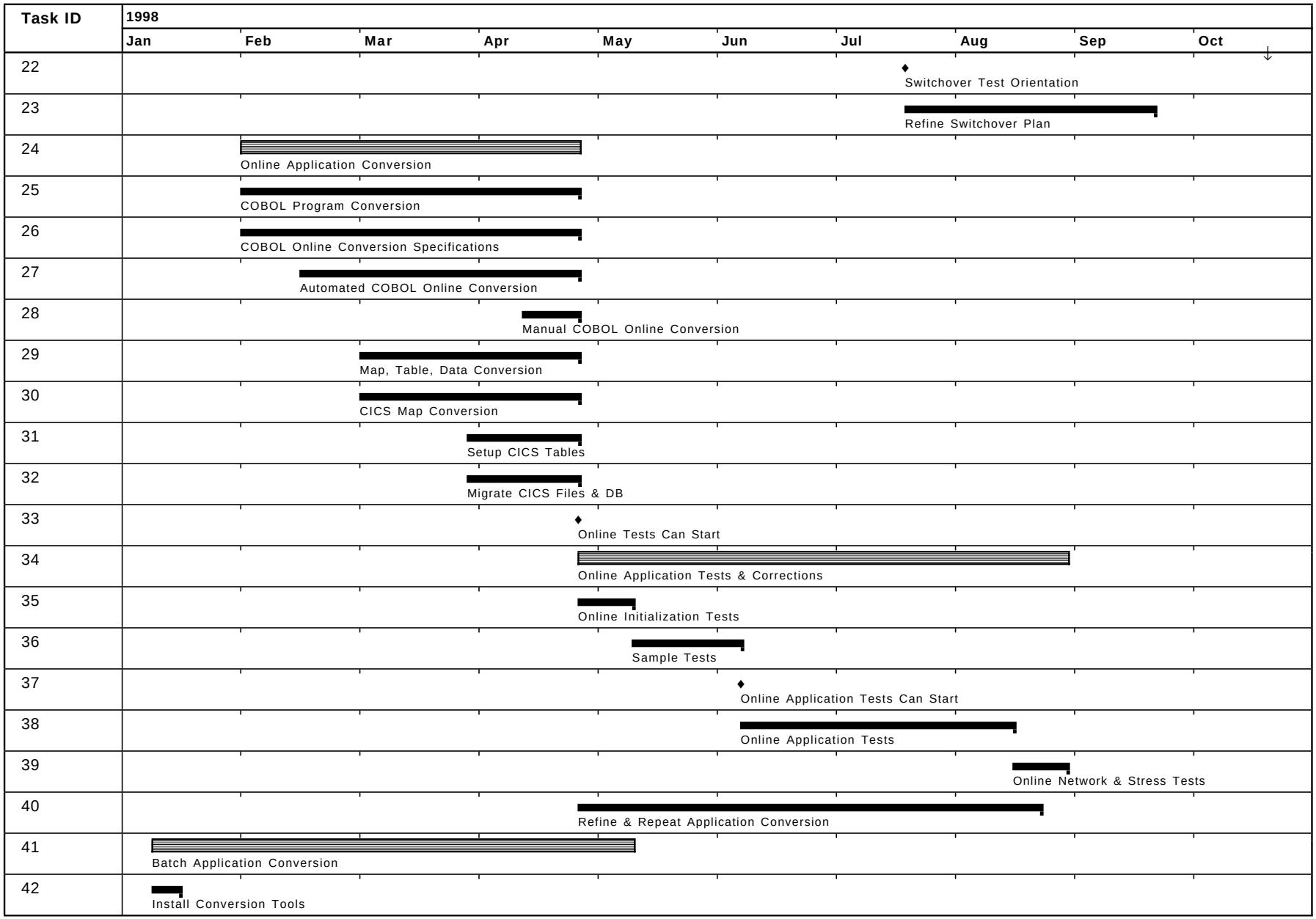
3.4.2.2 Project Plan - Details

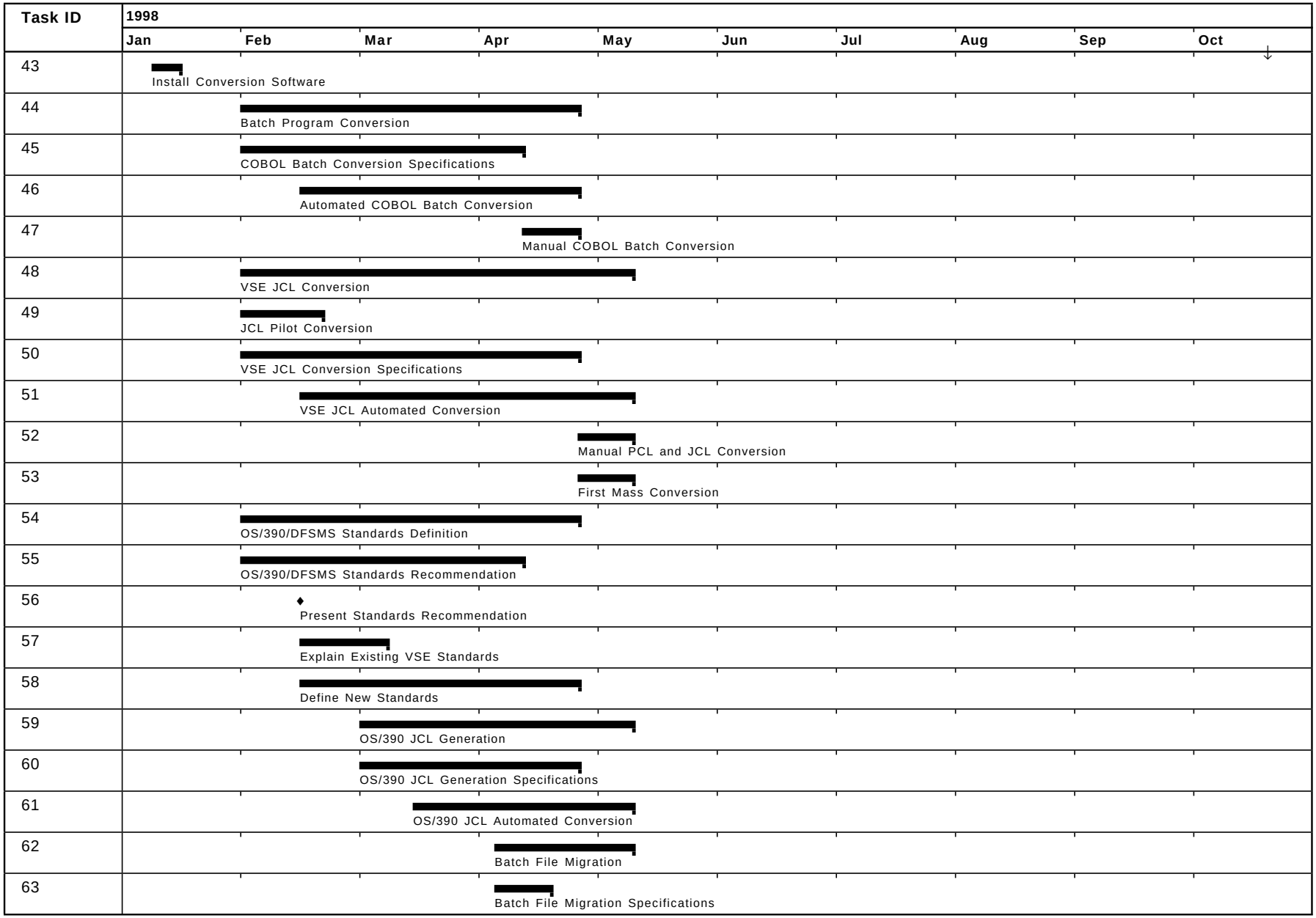
ID	Task Name	Projected		Actual	
		Start	End	Start	End
01	Project Approval			01/09/98	01/09/98
02	Application Inventory			01/09/98	08/24/98
03	Initial & Inventory Supplies			01/09/98	03/10/98
04	0% Missing			03/10/98	03/10/98
05	General Inventory Supply Every 3 Weeks			03/10/98	08/24/98
06	Less than 2% Missing			02/01/98	02/01/98
07	Project Planning	01/09/98	09/21/98		
08	Project Plan			01/09/98	02/13/98
09	Develop Project Plan			01/09/98	01/23/98
10	Present Project Plan			01/23/98	01/23/98
11	Review & Complete Project Plan			01/23/98	02/13/98
12	Online Test Plan & Scripts			02/14/98	04/26/98
13	Provide Online Test Orientation			02/14/98	02/14/98
14	Develop Online Test Plan & Scripts			02/15/98	04/26/98
15	Review Online Test Plan & Scripts			02/15/98	04/26/98
16	Batch Test Plan & Scripts			03/28/98	06/07/98
17	Provide Batch Test Orientation			03/28/98	03/28/98
18	Develop Batch Test Plan & Scripts			03/29/98	06/07/98
19	Review Batch Test Plan & Scripts			04/12/98	06/07/98
20	Switchover Plan			07/05/98	09/21/98
21	Develop Switchover Plan			07/05/98	07/18/98
22	Provide Switchover Test Orientation			07/19/98	07/19/98
23	Complete & Refine Switchover Plan			07/19/98	09/21/98
24	Online Application Conversion	02/01/98	04/26/98		
25	COBOL Program Conversion			02/01/98	04/26/98
26	Develop COBOL Online Conversion Specifications			02/01/98	04/26/98
27	Develop Automated COBOL Online Conversion			02/15/98	04/26/98
28	Perform Manual COBOL Online Conversion			04/12/98	04/26/98
29	Map, Table, Data Conversion			03/01/98	04/26/98
30	Migrate CICS Maps			03/01/98	04/26/98
31	Setup CICS Application Tables			03/29/98	04/26/98
32	Migrate CICS Application Files & Databases			03/29/98	04/26/98
33	Online Tests Can Start			04/26/98	04/26/98
34	Online Application Tests & Corrections	04/26/98	08/30/98		
35	Participate in Online Initialization Tests			04/26/98	05/10/98
36	Perform Online Sample Tests			05/10/98	06/07/98
37	Online Application Tests Can Start			06/07/98	06/07/98

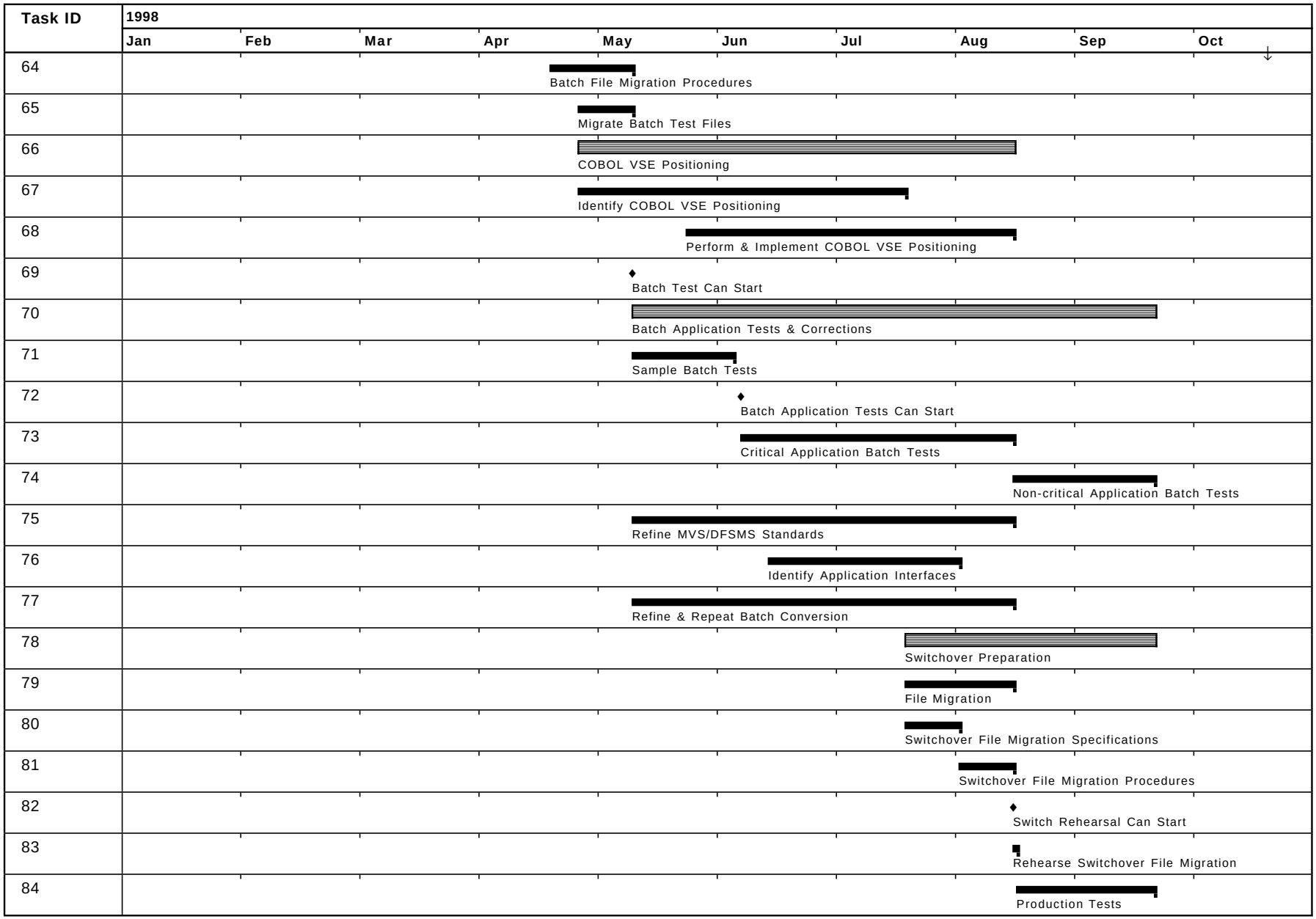
ID	Task Name	Projected		Actual	
		Start	End	Start	End
38	Perform Online Application Tests			06/07/98	08/16/98
39	Perform Online Network & Stress Tests			08/16/98	08/30/98
40	Refine & Repeat Online Application Conversion			04/26/98	08/23/98
41	Batch Application Conversion	01/09/98	05/10/98		
42	Install Conversion Tools			01/09/98	01/16/98
43	Install Conversion Software			01/09/98	01/16/98
44	Batch Program Conversion			02/01/98	04/26/98
45	Develop COBOL Batch Conversion Specifications			02/01/98	04/12/98
46	Develop Automated COBOL Batch Conversion			02/15/98	04/26/98
47	Perform Manual COBOL Batch Conversion			04/12/98	04/26/98
48	VSE JCL Conversion			02/01/98	05/10/98
49	Perform JCL Pilot Conversion			02/01/98	02/20/98
50	Develop VSE JCL Conversion Specifications			02/01/98	04/26/98
51	Develop VSE JCL Automated Conversion			02/15/98	05/10/98
52	Perform Manual PCL and JCL Conversion			04/26/98	05/10/98
53	Perform Initial Mass Conversion (JCL + PCL)			04/26/98	05/10/98
54	OS/390/DFSMS Standards Definition			02/01/98	04/26/98
55	Develop OS/390/DFSMS Standards Recommendation			02/01/98	04/12/98
56	Present OS/390/DFSMS Standards Recommendation			02/15/98	02/15/98
57	Explain Current VSE Standards			02/15/98	03/08/98
58	Define New OS/390/DFSMS Standards			02/15/98	04/26/98
59	OS/390 JCL Generation			03/01/98	05/10/98
60	Define OS/390 JCL Generation Specifications			03/01/98	04/26/98
61	Develop OS/390 JCL Automated Conversion			03/15/98	05/10/98
62	Batch File Migration			04/05/98	05/10/98
63	Develop Batch File Migration Specifications			04/05/98	04/19/98
64	Develop Batch File Migration Procedures			04/19/98	05/10/98
65	Migrate Batch Files for Testing			04/26/98	05/10/98
66	COBOL VSE Positioning	04/26/98	08/16/98		
67	Identify & Perform COBOL VSE Positioning			04/26/98	07/19/98
68	Perform, Test & Roll-out COBOL VSE Positioning			05/24/98	08/16/98
69	Batch Test Can Start			05/10/98	05/10/98
70	Batch Application Tests & Corrections	05/10/98	09/21/98		
71	Perform Sample Batch Tests			05/10/98	06/05/98

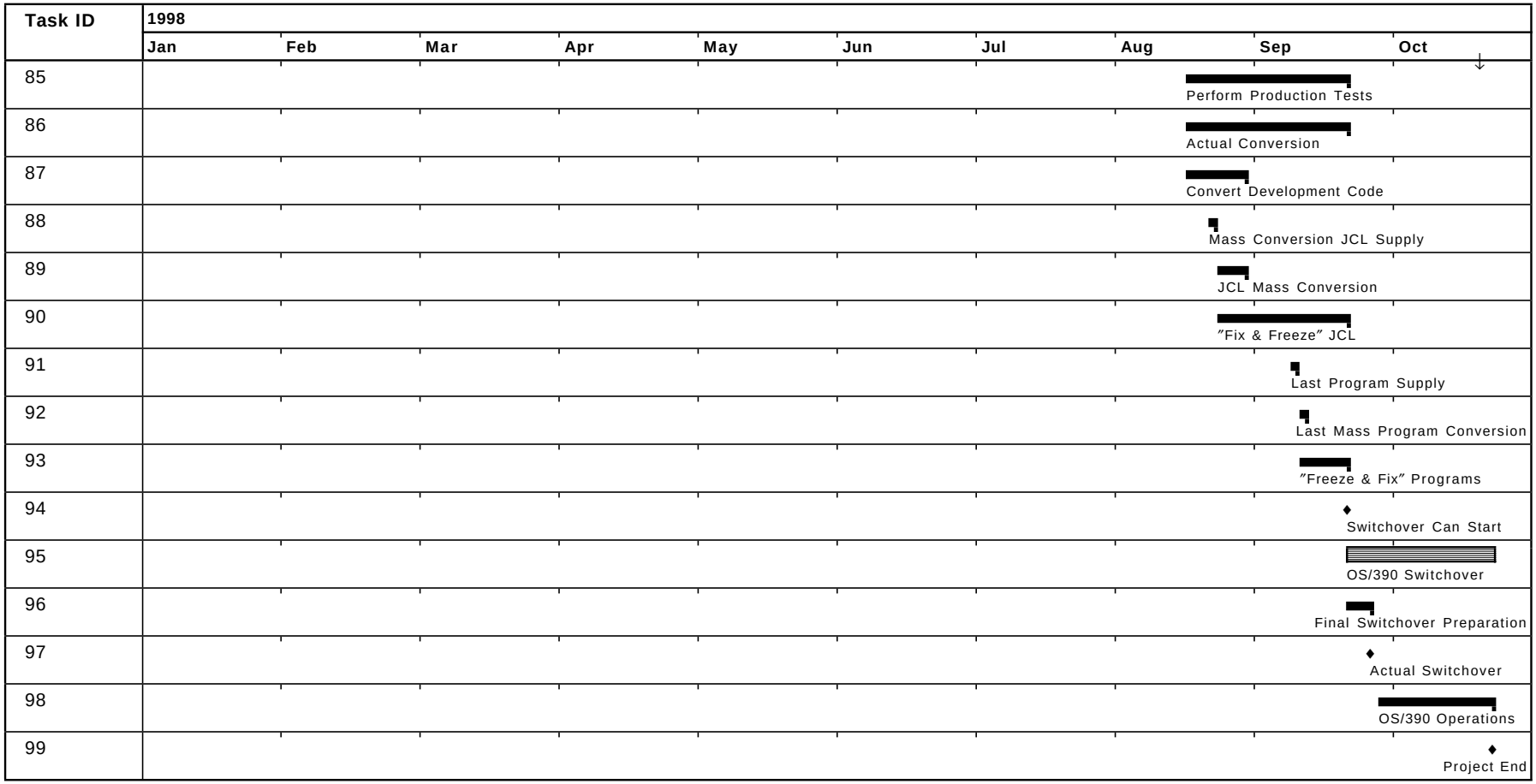
ID	Task Name	Projected		Actual	
		Start	End	Start	End
72	Batch Application Tests Can Start			06/07/98	06/07/98
73	Perform Critical Application Batch Tests			06/07/98	08/16/98
74	Non-critical Application Batch Tests			08/16/98	09/21/98
75	Refine New OS/390/DFSMS Standards			05/10/98	08/16/98
76	Identify Application Interfaces			06/14/98	08/02/98
77	Refine & Repeat Batch Application Conversion			05/10/98	08/16/98
78	Switchover Preparation	07/19/98	09/21/98		
79	File Migration			07/19/98	08/16/98
80	Develop Switchover File Migration Specs			07/19/98	08/02/98
81	Develop Switchover File Migration Procedures			08/02/98	08/16/98
82	Switch Rehearsal Can Start			08/16/98	08/16/98
83	Rehearse Switchover File Migration			08/16/98	08/17/98
84	Production Tests			08/17/98	09/21/98
85	Perform Production Tests			08/17/98	09/21/98
86	Actual Conversion			08/17/98	09/21/98
87	Convert Development Code			08/17/98	08/30/98
88	JCL Supply For Last Mass Conversion			08/22/98	08/23/98
89	Last Mass JCL Conversion			08/24/98	08/30/98
90	″Fix & Freeze″ JCL			08/24/98	09/21/98
91	Program Supply for Last Mass Conversion			09/09/98	09/10/98
92	Last Mass Program Conversion			09/11/98	09/12/98
93	″Freeze & Fix″ Programs			09/11/98	09/21/98
94	Switchover Can Start			09/21/98	09/21/98
95	OS/390 Switchover	09/21/98	10/23/98		
96	Final Switchover Preparation			09/21/98	09/26/98
97	Actual Switchover Weekend			09/26/98	09/26/98
98	Assist OS/390 Operations			09/28/98	10/23/98
99	Project End			10/23/98	10/23/98


Task ID
1998
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
01
Project Approval
02
Application Inventory
03
Application Inventory
04
0% Missing
05
General Inventory Supply Every 3 Weeks
06
Less than 2% Missing
07
Project Planning
08
Project Plan
09
Develop Project Plan
10
Present Project Plan
11
Review & Complete Project Plan
12
Online Test Plan & Scripts
13
Online Test Orientation
14
Develop Online Test Plan
15
Review Online Test Plan
16
Batch Test Plan & Scripts
17
Batch Test Orientation
18
Develop Batch Test Plan
19
Review Batch Test Plan
20
Switchover Plan
21
Develop Switchover Plan


Task ID
1998
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
22
Switchover Test Orientation
23
Refine Switchover Plan
24
Online Application Conversion
25
COBOL Program Conversion
26
COBOL Online Conversion Specifications
27
Automated COBOL Online Conversion
28
Manual COBOL Online Conversion
29
Map, Table, Data Conversion
30
CICS Map Conversion
31
Setup CICS Tables
32
Migrate CICS Files & DB
33
Online Tests Can Start
34
Online Application Tests & Corrections
35
Online Initialization Tests
36
Sample Tests
37
Online Application Tests Can Start
38
Online Application Tests
39
Online Network & Stress Tests
40
Refine & Repeat Application Conversion
41
Batch Application Conversion
42
Install Conversion Tools


Task ID
1998
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
43
Install Conversion Software
44
Batch Program Conversion
45
COBOL Batch Conversion Specifications
46
Automated COBOL Batch Conversion
47
Manual COBOL Batch Conversion
48
VSE JCL Conversion
49
JCL Pilot Conversion
50
VSE JCL Conversion Specifications
51
VSE JCL Automated Conversion
52
Manual PCL and JCL Conversion
53
First Mass Conversion
54
OS/390/DFSMS Standards Definition
55
OS/390/DFSMS Standards Recommendation
56
Present Standards Recommendation
57
Explain Existing VSE Standards
58
Define New Standards
59
OS/390 JCL Generation
60
OS/390 JCL Generation Specifications
61
OS/390 JCL Automated Conversion
62
Batch File Migration
63
Batch File Migration Specifications


Task ID
1998
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
64
Batch File Migration Procedures
65
Migrate Batch Test Files
66
COBOL VSE Positioning
67
Identify COBOL VSE Positioning
68
Perform & Implement COBOL VSE Positioning
69
Batch Test Can Start
70
Batch Application Tests & Corrections
71
Sample Batch Tests
72
Batch Application Tests Can Start
73
Critical Application Batch Tests
74
Non-critical Application Batch Tests
75
Refine MVS/DFSMS Standards
76
Identify Application Interfaces
77
Refine & Repeat Batch Conversion
78
Switchover Preparation
79
File Migration
80
Switchover File Migration Specifications
81
Switchover File Migration Procedures
82
Switch Rehearsal Can Start
83
Rehearse Switchover File Migration
84
Production Tests


Task ID
1998
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
85
Perform Production Tests
86
Actual Conversion
87
Convert Development Code
88
Mass Conversion JCL Supply
89
JCL Mass Conversion
90
″Fix & Freeze″ JCL
91
Last Program Supply
92
Last Mass Program Conversion
93
″Freeze & Fix″ Programs
94
Switchover Can Start
95
OS/390 Switchover
96
Final Switchover Preparation
97
Actual Switchover
98
OS/390 Operations
99
Project End

Part 2. Converting the VSE Operating System to the OS/390 Operating System

Chapter 4. Job Control Language (JCL) Differences and Considerations

The following sections describe the major tasks and considerations involved in converting VSE JCL to MVS JCL and the differences between them. These sections are divided into the following categories:

- 4.1, The Philosophy of JCL in System/390
- 4.2, High Level Similarities
- 4.3, JCL Differences Between VSE and MVS
- 4.4, JECL
- 4.5, VSE and MVS JCL Comparison Example

While this chapter describes the differences and conversion tasks, we recommend that you take a class on MVS JCL. See Appendix A, “Education Information” on page 535.

4.1 The Philosophy of JCL in System/390

Often, before discussing JCL systems and schemes, it is valuable to understand why the System/390 (originally the System/360) operating systems incorporated Job Control.

In the era of the predecessor computer systems, for example the IBM 1400, 7080, and 7090 systems, the concept of job control was just beginning. Application program coding included explicit references to files and other system resources. If a given program could be used with another file, the program often required changes. Flexibility and the beginnings of resource reuse led to the concept of a system facility that externalized the references from programs to other system resources, whether they were other programs or data files.

Job Control Language was developed as part of the System/360 architecture, to address the requirement for reuse. The ability to use one program with different files, and with different predecessor and successor programs, makes computer programs much more usable. This ability to create jobs and steps is crucial to the development of today′s ″Industrial Strength″ information processing technology.

As OS/390′s predecessors were being developed, it became obvious that the smaller customers′ needs required smaller systems. With the economics in the information processing over 30 years ago, the smaller systems were too small in terms of internal and external storage and processor power to provide the minimum environment needed for OS/360.

VSE/ESA′s predecessors were developed to permit smaller customers′ requirements to be met with the smaller systems then available. BOS (Basic Operating System), TOS (Tape Operating System), DOS (Disk Operating System), DOS/VS, VSE, and now VSE/ESA are the progression of operating systems designed to ″fill the hole″ left by small processor requirements that could not meet the minimum resource requirements of the OS/390 predecessors.

OS/360 (PCP, MFT, MVT), the predecessors to MVS, and OS/390, specified Job Control Language but when JCL was needed for BOS and TOS, a much smaller implementation was required. Different JCL philosophies developed from this background.

4.1.1 VSE/ESA′s Job Control Language Philosophy

Within the VSE/ESA philosophy for Job Control Language, several concepts are required:

- A ***Job Stream*** describes the concept of a single job or a sequence of jobs which must follow each other in sequence. These will run in a single address space or partition of the VSE/ESA system. They are delimited by ″Job Entry Control Language″, or JECL, which is interpreted by the VSE job spooling subsystem, POWER.

  Generally, sequencing of job streams is performed by the operator or by a job scheduling subsystem.

- A ***Job*** describes the concept of one or more job steps which relate the sequence of programs to be executed, together with the files and other system resources those job steps require for their successful execution. A job can be composed of one or more steps.

  Each job will have a known system initial state in terms of system resources, and at the end of the job, those resource assignments will be reset to their initial conditions. Thus, well-formed jobs can run independently with the exception of any input or output data files that they use.

  Execution of job steps is generally sequential, but the VSE conditional JCL facilities permit status checking and conditional or absolute GOTO capabilities, thus a given job will be able to modify its own processing sequence depending on results from earlier steps in that job.

- A ***Job Step*** is the smallest unit of job control from a scheduling perspective. Job steps receive the state as established by their predecessor steps, in terms of system resource assignment. Job steps are, for practical purposes, an instance of the execution of a single program, and specify the system resources needed by that program. They can affect resource assignment and state variables such as condition codes and parameter values for successor job steps, as well.

VSE Job Control is processed as job steps are executed. That is, no VSE system functions preprocess job control statements for syntax or resource availability checking before the actual execution of the statements. In addition, VSE provides for standard resource assignments and file definitions through the concepts implemented as ″Permanent ASSGNs″ and ″Standard Labels″, which can be used by any job or step without any specific inclusion in the JCL defining that job or step. These capabilities make VSE JCL less complex to code, but more complex to understand, as it is interpreted at step execution time in the context of the permanent ASSGN and standard label environment in place at that time.

4.1.2 OS/390′s Job Control Philosophy

The concept of a job stream, that is a collection of related jobs, does not exist in OS/390 and JES2. If a group of jobs is to flow in a particular sequence, you can create a single new job with the same sequence of job steps. You can also use a job scheduling product such as OPC which can control the flow and sequencing of multiple jobs.

Because there is no concept of permanent ASSGN or specification of standard label facilities, all resource requirements for each job step must be included explicitly in each job step. In OS/390, these definitions can be included as procedures which can reduce the repetitive coding of JCL statements (specifically DD statements).

To understand the structure and philosophy of MVS job control language, you must first understand the workflow process for batch jobs. With OS/390, there are separate phases for each of the following:

1. Input Service

   The job's JCL and any instream data sets are read by JES and stored on spool as related spool files. No procedures or included JCL statements are referenced or verified at this time.

   JES control statements (JECL) are read, validated and attributes are recorded for later processing. (These are converted to JCL comment cards so the subsequent stages do not process them.)

2. JCL Conversion

   The job control statements are "converted" into structured text units, and most (but not all) syntax checking is performed. This step usually takes place immediately after input service, and any JCL errors result in the job being queued for output processing, bypassing execution, with a JCL error message being sent to the submitter.

3. Job Scheduling

   After conversion, the job is queued for initiation, and selected by either a JES2 or WLM managed initiator on a priority, class, and first-come, first-served basis.

4. Job and Step Initiation

   When the job is selected for initiation, the converter-interpreter text units are read and any data set references are resolved. Any errors such as "data set not found" are determined at this point and the job is flushed - queued for output processing - and the programmer is notified.

   Once the interpreted control blocks are read into memory, each job step is given control one at a time based on the conditional JCL processing options. At step initiation time, all data sets referenced by JCL statements are allocated. This is unlike VSE when data sets are allocated when they are opened.

5. Output and Hardcopy Processing

   There are actually two steps here with JES2. When the job finishes execution, the output created on JES spool is grouped into output elements according to destination and similar attributes, and queued for hardcopy processing.

   These output elements are then individually scheduled for printing, punching, online viewing, or transmission to another node.

6. Purge Processing

   After all the output for a job is disposed of, the job is deleted from the JES queues and the spool space is freed up.

Unlike VSE, the operator does not manipulate elements within a job stream, nor is he given the opportunity to correct JCL errors. The processes are much more automated in OS/390 under the theory that the system will be better utilized and jobs run more efficiently without operator intervention.

4.2 High Level Similarities

A high level comparison of JCL in the VSE and OS/390 environments reveals many similar functions and purposes. A comparison of the mechanics in both environments reveals significant differences.

4.2.1 JCL Statement and Job Layout

VSE and MVS JCL are similar in the basic layout for the card images in that both use 80 Column Card Images, and both use // in columns one and two. Both operating systems also use the basic layout of a job with one or more steps per job as described in the philosophical discussion above.

4.2.1.1 Continuation Cards

Both use ASM-type continuation, but the basic layout differs in that:

- VSE JCL statements are continued by placing a non-blank character in column 72, and JCL continuation cards must start in column 16 with blanks in columns 1 - 15.
- MVS JCL statements are continued by placing a trailing comma in the parameter field, and JCL continuation cards may start in any column from 4 to 16, with ″//″ in columns 1 and 2, and a blank in column 3.

4.2.1.2 JOB Statement Starts a Job

In OS/390 there is only one JOB statement as opposed to the VSE POWER and VSE JOB statements. Much of the time the POWER job will equate to the MVS job.

4.2.1.3 EXEC Defines Job Step

The EXEC statement defines the job step in both VSE and MVS.

4.2.1.4 File Definitions

File definitions are required by both operating systems (TLBL, DLBL/EXTENT, DD).

4.2.1.5 Imbedded JCL from Procedures and Libraries

Both operating systems support “canned JCL” and JCL procedures. In VSE, this is done through procedures (using PROC=procname in the EXEC card) and with the POWER * $$ SLI JECL statement (Source Library Inclusion). In OS/390, the same thing can be done with the PROC or INCLUDE JCL statements, respectively.

4.2.1.6 Nesting Procedures

Both operating systems allow for multiple levels of nested procedures. (MVS allows up to 15 levels while VSE allows up to 16 levels.)

4.2.1.7 Instream Data

Both operating systems allow Instream Data in the middle of the JCL. This is data that will be processed by the executing program.

4.2.1.8 Variables

The JCL in both operating systems will accept variables. These variables are set with the // SET statement in MVS, or SETPARM in VSE.

4.2.1.9 Conditional JCL

Conditional JCL exists in both environments and allows performing IF and GOTO statements. Loops are prohibited. In MVS, the IF statements are all processed at converter time. Although the mechanics are very different, functionally, the IFs are the same in both environments.

4.2.1.10 Return Codes

Return codes from previous steps can also be tested during execution in both environments. Program steps can be bypassed based on the result of testing for a condition (a return code or a parameter value). For example, if in a statement, the return code was more than zero, then bypass the next statement. In MVS, this is handled by the COND parameter on the // EXEC statement.

See also 4.3.11.3, “MVS Conditional JCL” on page 84.

4.2.2 Spooling

Spooling exists in both environments. POWER for VSE and JES for OS/390. POWER and JES provide similar input and output capabilities and a similar system of classes and priorities.

4.2.2.1 Internal Reader

The internal reader facility exists in both environments. An application program can pass jobs to the spool, right into the input queue, just by writing to a pseudo punch device. RJE and NJE also exist in both environments.

Further discussion of spooling can be found in Chapter 10, “POWER and JES2” on page 207.

4.3 JCL Differences Between VSE and MVS

In part because of the differences in philosophy discussed in 4.1, “The Philosophy of JCL in System/390” on page 69, there are differences in the processing of JCL that lead to Job Control Language differences between the two environments.

4.3.1 Job Input

In VSE systems, job input consisting of JCL statements and instream data, whether spooled through the POWER spooling system or directly read in a partition, is processed in a strictly sequential process. That is, a program can only read one input statement at any one time, and this is a sequential process. A programming or JCL error in VSE can cause the VSE Job Control program to read a user data statement, or a user program to read a JCL statement, with unpredictable results.

Instream data will always follow an EXEC statement, and it is the responsibility of the executing program which is reading the instream data to recognize the end of that data. By default, the instream data delimiter is the ″/*″ statement, although an application program can choose its own delimiter. This allows programs other than JCL to read and process JCL statements, for example, when the librarian program stores JCL as library members or procedures.

This same capability was often used to control the flow of jobs -- for example, a program could decide to skip the next job step, and then just read and ignore (or ″swallow″) the JCL statements for that step. With the advent of VSE conditional JCL in the mid-1980s, the use of this technique has greatly declined, but its use is found in perhaps 25% of shops converting from VSE to OS/390.

In MVS systems, in contrast, JCL statements and instream data are separated during the JCL Conversion processing, so that user programs cannot ″see″ JCL statements, and JCL processing is simplified.

Instream data sets in the OS/390 environment can be read in any sequence, and can be read multiple times. Thus, an OS/390 job that reads the same instream input at three different times could simply open and process that data set three times.

4.3.1.1 Multiple Instream Data Set Input

A VSE job step that reads one input card file under two different program DTFs requires that the input statements be properly sequenced, whereas in OS/390, the two input files could appear as two separate instream files.

VSE Example

```
// EXEC MYPROG...
    FILE 1 CARD 1
    FILE 1 CARD 2
    FILE 2 CARD 1
    FILE 1 CARD 3
    FILE 1 CARD 4
    FILE 2 CARD 2
    FILE 1 CARD 5
    FILE 1 CARD 6
    FILE 2 CARD 3
/*
```

OS/390 Example

```
//FILE1 DD *
    FILE 1 CARD 1
    FILE 1 CARD 2
    FILE 1 CARD 3
    FILE 1 CARD 4
    FILE 1 CARD 5
    FILE 1 CARD 6
/*
//FILE2 DD *
    FILE 2 CARD 1
    FILE 2 CARD 2
    FILE 2 CARD 3
/*
// EXEC PGM=MYPROG...
```

For this processing to work correctly in VSE, it is clearly dependent upon the program logic and the setup of the instream data. This would be much simpler in the OS/390 environment. If the MVS example attempts to use just one instream data set, with two program files being read, each program file will find the same input data. That is, the first read (from file 1) would read the first record, and the second read (from file 2) would also read the same first record, as it is the first read for that file.

4.3.1.2 Data Driven Segmentation of Output

An artifact of this sequential processing in VSE is that the spooling system extracts its control statements (JECL) from the input as it is being spooled. When the input processing crosses the input stream position where the JECL statement was located, the spooling system will take the action specified by the JECL statement. The JECL statement can change the output destination of printed or punched data, or other characteristics, such as special forms requirements.

VSE Example

```
* $$ JOB JNM=DJANDA,CLASS=A
* $$ LST CLASS=A,DEST=*
// JOB DJANDA
// EXEC MYPROG
INPUT DATA CARD 1
INPUT DATA CARD 2
INPUT DATA CARD 3
INPUT DATA CARD 4
INPUT DATA CARD 5
* $$ LST CLASS=J,DEST=DANJ,DISP=H
INPUT DATA CARD 6
INPUT DATA CARD 7
INPUT DATA CARD 8
INPUT DATA CARD 9
INPUT DATA CARD 10
INPUT DATA CARD 11
/*
/&
    * $$ EOJ
```

The result would be that output printed by this job and program would be sent to the default system printer, up to the point when the program read INPUT DATA CARD 6. Output generated from that point forward (including that of cards 6 through 11) would be sent to a specific user ID, DANJ, rather than the default printer, and it will be in the HOLD disposition state.

A second type of input segmentation appears when a given program will open an instream data file, read some of its records and close the file. Later, it will open the same instream data file and read additional records. In VSE, the records read in the second group will follow the first group in the input stream. A simple conversion to OS/390 will result in the second file open re-reading the same records read by the first file!

A method to circumvent this problem is to change the program logic or to write a subroutine which traps all the reads on the two input streams and which has one single DCB, so there is only one DDNAME and then the behavior would be similar to the VSE case.

4.3.1.3 JCL Parameter Handling

Another difference seen because of the philosophy and architecture changes between VSE and OS/390 is the fact that VSE JCL parameters and JCL handling can depend on values that are changed at execution time. VSE conditional JCL can test return codes, as MVS JCL can, but in addition, VSE can test parameter values as well.

Also, in VSE, procedure expansion and parameter substitution is done at execution time, so the results of previous job step execution can affect the

expansion for subsequent steps. In general, this is not possible in MVS JCL in the OS/390 environment.

4.3.2 JCL Expansion

In VSE, JCL is expanded at execution time. The most current changes, even those changed two seconds before the job begins execution are included. The first step could change a procedure that is used by the third step.

In OS/390, JCL is expanded all at once when the job is submitted. This may be hours or days before it is executed. All the procedures, all of the jobs, all of the includes that are required are expanded at the same time. You can not change variables during execution in OS/390 using symbolic names in JCL.

If a job is submitted today to be run tomorrow and overnight one of the included members is changed and not reflected in the original JCL that was submitted, the job will fail.

4.3.2.1 Early Error Detection

An advantage of expanding the JCL when the job is read in OS/390 is that many of the JCL errors will be detected early and the job will fail. This removes the ability to correct the job on the fly as in VSE. In OS/390 many errors are detected before the job starts. In VSE a card has to be processed before errors are found and the operator can act on it. However, because of this, you must have a syntactically correct JOB statement to get JCL errors from OS/390.

4.3.2.2 Overrides

The OS/390 ′Overrides′ occur at the step level. A procedure can only be overridden in OS/390 through the addition of a DD Statement to a specified step. This is different from VSE, where statements in PROCs and SLIs are overridden using each statement′s sequence number in positions 73-80.

4.3.3 Operator Flexibility and Intervention

Another difference between VSE and MVS JCL is the flexibility created by requiring operator intervention. For example, the operator may correct invalid JCL syntax.

4.3.3.1 Correcting Invalid Syntax

A syntax error in a JCL statement will cause a message to be posted on the operator console. The operator will respond to the message by typing in the correct JCL statement and processing will continue. This facility is not available with OS/390. The job will fail with a JCL error, and must be corrected and resubmitted.

4.3.3.2 Operator Data Entry

From a VSE point of view this flexibility is a feature. Installations depend on this flexibility to address situations where it is necessary to have the operator retype the JCL or command. For example, a programmer may purposely put in a syntax error in the JCL to ensure it comes to the operator′s attention. The experienced operator retypes the string and allows the job to continue.

This technique is illustrated in the following example, where the syntax of the ASSGN statement is invalid and causes the operator to be prompted for action:

```
// PAUSE  mount tape 123456 on an available drive.
// ASSGN SYS005,CUU
       - or -
// ASSGN SYS005,Drive
```

The operator gets an error message and enters the correct tape assignment, such as:

```
// ASSGN SYS005,481
```

Another example is TLBL without VOLSER or a known invalid VOLSER.

```
// TLBL 999999
```

Forcing error conditions in VSE JCL that require operator intervention becomes a simple tape management method.

IGNORE, DELETE and NEWTAP are replies that the operator can use to answer the messages.

IGNORE IGNORE can be used when the data set name in the JCL does not match the data set name on the tape. IGNORE says go ahead and read it anyway. It acts as a BLP (Bypass Label Processing) in OS/390. This allows tapes with any labels or no labels to be processed without label verification.

Another approach is to leave the data set name blank which allows whatever is on the drive to be read.

DELETE DELETE is used to create a new file disk when files already exist in these extents. Using DELETE will overlay the previous extents. DELETE can also be used to overwrite extents when someone has used your extent.

NEWTAP NEWTAP is used to unload the volume currently mounted on the drive and allow the operator to mount another volume.

4.3.3.3 Comment Lines in the JCL

Comment lines can be inserted in the VSE JCL, wherever a valid JCL is allowed. Comment lines that start with an asterisk are displayed on the console and, if option LOG is set, the message is also written to SYSLST.

To have comments in the VSE JCL not written to the console use ′/*′ instead of ′*′. Exercise caution as to where they are placed because ′/*′ still means end of data. Some people also accomplish this by using ′/.′. What is important is that the first word must be a valid JCL label.

The MVS JCL offers no possibility to have the comments written to the console; comment lines are only written to the job′s output.

4.3.3.4 PAUSE Statement

The PAUSE statement provides the ability to halt job execution and wait for operator intervention. VSE comment lines (with an asterisk in position 1) can be used before a PAUSE statement to display multi-line messages to the operator. See 4.5.1, “Sample VSE JCL” on page 92 where there is an example of a multiple line message being sent to the console with a comment line and a PAUSE to send a message to the operator and wait.

There is no equivalent function in OS/390. Many users write their own routine to replace the PAUSE function, if needed, or use the existing automation functions of OS/390.

4.3.4 Allocation of Resources

The allocation of resources in OS/390 occurs at step initialization time. This is a big difference from VSE where allocation of resources occurs at open time. In OS/390, if the JCL contains DD statements that point to data sets, the data sets are allocated even if they are not opened. This makes IEFBR14 useful in testing out allocations or in allocating new data sets.

In MVS, IEFBR14 can be used to run a job with no program execution. This causes all JCL to go through conversion and interpretation. Using IEFBR14 in VSE would not cause a file to be opened, so no allocation of resources could take place.

With MVS, the system allocates the resources when a job step is started. The volumes have to be mounted, the devices have to be available, the system data sets have to be there and the region has to be available. If there are data sets in your JCL, they have to be cataloged and the volume information must be specified. If more tape drives are required than are available to execute the job, the job will wait until sufficient tape drives are made available.

With MVS, the scope of allocation is generally the current step. JCL statements, such as the DD statement, affect only the current job step. This is very different from VSE where the ASSGN statement can secure a tape drive for the duration of the job. In OS/390, a tape drive deallocated at the end of a step may be ″stolen″ by another JOB before the beginning of the next step, causing the tape to be dismounted and a mount message to be issued on another tape drive.

4.3.4.1 Resource Allocation at Open Time

With VSE JCL, allocation of resources is done at OPEN time. The JCL can contain numerous DLBL or TLBL statements, even for files that do not exist; as long as the application does not open them, they′re just ignored. This is completely different in OS/390 where files specified in JCL are allocated at step initiation time, whether the application opens them or not. See item 4 on page 71.

4.3.5 Hidden JCL

Hidden JCL doesn′t appear in the job stream but is used during the execution of the job. It can be the permanent assignments, labels in the partition or system standard label areas, or ″carryover″ (described below).

4.3.5.1 Partition and System Standard Labels

File definitions stored in the partition and standard label areas do not appear in the VSE JCL. In OS/390 all file definitions are coded in the JCL.

System standard labels provide a set of labels that is common to the whole VSE system.

Partition standard labels provide a set of partition specific file definitions. These file definitions are different from one partition to another. This function can be used for application, sort or compiler work files. This may have an impact on

the conversion of the JCL which may have to be converted differently, depending on in which partition the job normally runs.

There is no equivalent to standard labels in OS/390. Labels must be spelled out in each step. One way to keep a common set of labels in OS/390 is to use the JCL INCLUDE statement. In rare cases another possibility is to use dynamic allocation to replace standard labels.

4.3.5.2 Permanent Assignments and POWER Defaults

Readers, punches, printers or disks can be permanently assigned in VSE. These assignments do not appear in the individual job JCL. OS/390 does not use the concept of a permanent assignment.

Spool devices can be started at POWER initialization time. They can also be stopped or started by the operator from the Attention Routine. This also does not appear in the JCL.

4.3.5.3 ″Carry-Over″

Carry-Over is a user term for a phenomenon that exists in VSE JCL and is otherwise undocumented. Carry-over occurs where a job step has tape or disk file definitions with DLBL or TLBL and there are additional execute statements. These execute statements will have access to these file definitions which are in a previous step. This can occur as long as this is not a new job and there are no intervening DLBL or TLBL (see 4.5.3, “Sample VSE plus Carry-Over” on page 94).

One common method in VSE is to put all the file definitions at the top of the job and then all that remains is the execute statements. The problem this creates is with translation, as it is difficult to associate the program with the file it uses.

The conversion task is to translate the programs to identify what files are used. The output is to produce a table or listing that has the file information and then merge the information with the JCL. It is a process of submitting all job steps and removing one level after the other to see which one is used and which one is not.

The opposite is also true in that some of the JCL may never be used. TLBLs and DLBLs may be present that are never used by that job or are used infrequently.

4.3.5.4 Help for the Hidden JCL Problem

Recent releases of VSE/ESA (Version 2.2 and later, and Version 1.4 and 2.1 with maintenance added) provide additional functions which can help with the Hidden JCL problem. The availability of Opti-Audit for VSE (also available from Barnard Software, Inc. as well as a part of the VSE/ESA Version 2.2 system) provides the ability to track usage of VSE program resources by job and job step, and of files by program or job step and by job. Opti-Audit can also collect inventory data from a running VSE/ESA system including the resources allocated through Standard Labels, Permanent Assignments, POWER defaults, and carry-over.

Another new function is called the VSE/ESA JCL Analyzer, which consists of a group of programs and files which gather file usage information from a running VSE/ESA system, and use that data to create an output file suitable for processing by the VisualAge tools on a workstation. The Application

Understanding tool provides a graphical analysis of your VSE/ESA JCL job stream. You can find further details at the following World Wide Web sites:

```
http://www.software.ibm.com/ad/va2000
http://www.software.ibm.com/ad/cobol
```

The JCL Analyzer is shipped as part of VSE Central Functions in VSE/ICCF library 59. It consists of a number of members, including ARDWREAD, which is a detailed description of the JCL Analyzer and its functions. All the other related library member names begin with the characters ARDW.

There is a brief description of this new function in the manual, *VSE/ESA Enhancements Version 2 Release 3*, SC33-6629

4.3.6 Device Address Specifications

In VSE, the Logical Unit Address, is the symbolic link between the program and the external units (tape drive, printer, and so on) it uses. The Logical Unit Address is a name in the form of SYSnnn, such as SYS004 or SYSLST. The Logical Unit Address is specified by the programmer on the DTF using the DEVADDR=SYSnnn keyword; for this reason, it is often referred to as the ″Device Address″, a term easily confused with ″Unit Address″, which refers to the external unit associated with the Logical Unit Address, such as the 3205 printer at address 00E.

In other words, the terms Logical Unit Address and Device Address both refer to the SYSnnn name, where Unit Address refers to the hardware device at address CUU.

In the VSE JCL, the ASSGN statement is used to associate a Device Address to a Unit Address; for example:

```
// ASSGN SYS010,FEF
```

where SYS010 is the device address specified in the program, and FEF is the unit address of a real or virtual printer device.

An ASSGN statement is normally required for every non-disk file used by the program, although Tape Management Software products generally remove the requirement for tapes.

There is no exact equivalent for the Device Address in MVS JCL. The association between a file and a particular device is established by Device Allocation, a system function invoked by the initiator. The equivalent of the above ASSGN card depends on what DDNAME is used in MVS for a particular file, which could be, for example:

```
  //SYS010 DD SYSOUT=...
or
  //REPORT1 DD SYSOUT=...
```

One may wonder, when converting VSE JCL to MVS, what to do with the Device Addresses, ASSGN cards, and what DDname should be used for card and print files. Here are some guidelines:

- For disk - the DEVADDR should be ignored, and the DTFname should be used as MVS DDname.
- For labeled tape - the DEVADDR should be ignored, except when the ASSGN statement specifies a tape density, or when assigning several files to the same unit. The DTFname should be used as MVS DDname.

- For unlabeled tapes - the DEVADDR is the only link between the program and the JCL (there is no TLBL). Either the DEVADDR or the DTFname can be used as DDNAME.
- For card devices - the DEVADDR links the DTF to a card reader or puncher; Either the DEVADDR or the DTFname can be used as DDNAME.
- For print devices - the DEVADDR is the only link to a printer or a LST card. Either the DEVADDR or the DTFname can be used as DDNAME.

4.3.7 Catalogs

With the exception of VSAM, in VSE JCL there are no catalogs to deal with. The user must provide an extent for each read or write to a disk file.

For tape functions the correct tape must be mounted. Volume information must be specified for disk files using an EXTENT card, except for files managed by VSAM, which are managed in a VSAM catalog.

There are vendor products in the marketplace that provide these cataloging functions, such as Dynam and EPIC. Many VSE installations use one of these products for disk management, tape management, or both. Installations that do not use a third-party disk manager often make extensive use of VSAM-managed SAM files.

4.3.8 Partition Dependent Codes in JCL

4.3.8.1 Procedures

Partition-dependent codes in VSE JCL can ensure that a procedure runs in a particular partition. Procedures may be cataloged names in the form of $xABC (where x = 1, 2, 3,... B = representing BG, F1, F2, F3,... FB partitions). A job may be built with an EXEC PROC=$$ABC and run in various partitions. When run in BG, $0ABC.PROC will run; when run in F5, $5ABC.PROC will run; and so on. MVS, which has no notion of ″partition″, has no equivalent function.

4.3.8.2 Data Set Names

Data set names can contain the condition dependent operands; *′%%′*. The first *′%′* is partition, the second *′%′* is the view.

This function is similar to the DSN=&&dsname function in MVS, which allows use of the same JCL in concurrently running jobs without having conflicts with the data set names.

4.3.9 Communication Region - DATE and UPSI

4.3.9.1 DATE

In VSE, the date is stored in the Job Date field of the partition′s communication region. There is only one facility in VSE from which to get the date. The date can be entered in the JCL and is what the job will see whenever the application queries the system date. The result will be the date that is specified in JCL.

The VSE DATE function allows a job to run with a date that is not the system date. It can also ensure that when a job that takes an hour to run is started at 23:30, the steps that execute after midnight will maintain the same date.

In OS/390 the DATE function can be replaced by a control card, a parameter on the EXEC statement, or a date simulation tool.

4.3.9.2 UPSI

The UPSI switches that were on the 1401s got a second life in DOS with the System/360. UPSI can be tested in RPG, Assembler and in COBOL. For more information on the manipulation of UPSI with Assembler see Chapter 13, “Assembler” on page 267. In OS/390, COBOL and RPG support UPSI through the PARM= on the EXEC statement. There is no support for UPSI in Assembler or PL/I. A feature of VSE UPSI statements is that they are carried over from one step to the next until the end of the job.

Many VSE utility programs that use UPSI have an equivalent in MVS which, obviously, does not use UPSI. For this reason, it is frequent that a large number of UPSI cards in the VSE JCL do not need to be converted to MVS.

The VSE application can modify the value of the UPSI byte internally using the MVCOM macro. This can be identified by inspecting the MVCOM macros in Assembler subroutines.

4.3.10 VSE Job Control Statements

4.3.10.1 Job Statement

The job statement is mandatory in OS/390 (it could be omitted in VSE; some say this makes it optional). Many times the VSE job card may be used to delineate a step (that is, if there is only one EXEC statement in the VSE job being converted).

Accounting information from the VSE job card may be specified in the MVS JOB or EXEC statement using the ACCT= keyword.

4.3.10.2 EXEC Statement

The EXEC statement in VSE is similar to MVS′ EXEC statement. It is used to identify the program or procedure to be executed. It also specifies storage SIZE requirements (similar to the MVS EXEC REGION parameter), and parameters to be passed to the program or procedure to be executed.

There are differences in defaults and parameter specifications. In VSE, the default for the name is a program module, while in MVS, the default is the name of a procedure. Thus, in VSE, you find

```
// EXEC PROC=procname,..    to execute a procedure
// EXEC IDCAMS,...          to execute a program
// EXEC REXX=rexxproc,..    to execute a REXX procedure
```

while in MVS you would find

```
//STEPNAME EXEC procname,...        to execute a procedure
//STEPNAME EXEC PGM=IDCAMS,...      to execute a program
//STEPNAME EXEC PGM=IRXJCL...       to execute a REXX procedure
```

4.3.10.3 TLBL Statement

The TLBL in VSE is equivalent to the DD statement for a labeled tape file in OS/390 (an unlabeled tape does not need a TLBL). The VSE *filename* (the DTF name) which can be up to seven characters long, is equivalent to the MVS DDname, which can be up to eight characters long.

The VSE *file-id* (the label), which can be up to 17 characters long, is equivalent to the MVS DSname, which can be up to 44 characters long.

4.3.10.4 MTC Statement

Magnetic Tape Control statements provide control over tape processing including writing tape marks and unloading tapes. MTC has no direct equivalent in MVS: OPEN automatically positions the tapes based on the LABEL parameter of the DD statement and CLOSE rewinds or unloads volumes, depending on the DISP parameter. Writing a tape-mark (MTC WTM) can be achieved with IEBGENER, as follows:

```
//WTM     EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DUMMY,RECFM=F,BLKSIZE=80
//SYSUT2   DD UNIT=TAPE,LABEL=(,NL)
```

4.3.10.5 ASSGN Statement

In VSE, the relationship between a logical unit (SYSxxx) used by a program, and a physical device used to contain a file is established by the ASSGN JCL statement. These assignments can be made temporarily -- that is, they will revert to a standard value at the end of a VSE JOB or when a RESET statement is processed, or they can be made permanent, where they will become the new persistent value.

Often ASSGN standards are established during system initialization and JCL will not explicitly repeat those ASSGN statements. Further, in the case of VSAM files, ASSGN processing is handled automatically by VSAM without the need for any ASSGN statements.

In MVS JCL, the closest analog is the use of the UNIT= parameter on a DD statement.

4.3.10.6 RESET Statement

The RESET statement resets the current temporary ASSGN value and the assignment of the logical unit will revert to its current permanent value.

There is no equivalent of this function in MVS JCL in OS/390, as there is no persistence or ″carry-over″ of device allocations from one job step to another.

4.3.10.7 DLBL and EXTENT

The DLBL and EXTENT statements provide information for disk files. The DLBL provides information such as the filename (7-character name), the file-id (1-44 character name), retention period, file type (VSAM, SD, DA) plus more. The EXTENT provides volume information and extent information for new data sets. VSAM files don't require EXTENT statements. The VSE *filename* is equivalent to the MVS DDname and the VSE *file-id* is equivalent to the MVS DSname. The *filename* (DSname) is the name written to the VTOC.

4.3.10.8 CAT=Catalog on DLBL

Another of the VSE differences is that the catalog is specified on the DLBL statement. It is similar to having a //STEPCAT for every DD statement for VSAM.

Each disk file can have its own catalog pointer. This provides the ability to have VSAM files that have the same name in different catalogs. When these files having the same name are migrated, consideration must be given to the target

location for these. In OS/390 these duplicate names need to be resolved somehow. Refer to Chapter 5, “Disk and Tape Storage Considerations” on page 97.

4.3.10.9 Conditional JCL

VSE/ESA added conditional JCL processing in the early 1980s, with VSE/SP Version 2. This conditional JCL is characterized by several new JCL statements, including // IF, // GOTO, // ON, and /. LABEL statements which allow the user to make JCL decisions based on the results of conditions set by previous JCL statements and program executions during the job.

The // IF statement can test the value of the last return code, the maximum return code found so far during this job, or the value of parameters. Based on the test, the subsequent statement can be skipped or executed. The // GOTO statement, in conjunction with the /. LABEL statement, allows steps or groups of steps to be skipped. A // GOTO statement will only skip in the forward direction -- looping is not allowed.

4.3.11 MVS Job Control Statements

4.3.11.1 DD Statement

The main JCL statement in MVS is the DD statement. It replaces the VSE JCL TLBL, DLBL, ASSIGN, RESET and EXTENT statements. Because it replaces so many JCL commands in VSE it can become complex. One complaint from VSE users is a DD statement must be coded for every step of every job where a file is used. If a change is made to a data set name, the JCL must be changed everywhere the file is used. In VSE these labels may be located in Standard Labels where they can be changed in one place.

4.3.11.2 OUTPUT JCL Statement

The OUTPUT JCL statement was added to JES in release 2.1.3. This replaces the ′/*OUTPUT′ JECL statement which provides various output attributes. With the ″DEFAULT=YES″ parameter, the OUTPUT JCL statement can provide step level or job level default attributes, much like the JOBLIB, STEPLIB and STEPCAT statements.

See 4.4, “JECL” on page 89 for more details.

4.3.11.3 MVS Conditional JCL

You can conditionally execute steps in a job by using the IF/THEN/ELSE/ENDIF statement construct or the COND parameter.

IF THEN ELSE ENDIF Statements

Depending on the results of a job step, you might need to bypass or execute later steps. For example, if a step terminates abnormally, you might want to execute an error routine procedure; while if the step terminates normally, you want to continue processing with the next step.

The conditions can be based on return codes, ABEND codes, or whether the job step ran or not.

COND Parameter on the EXEC Statement

To indicate the results of its execution, a program can issue a return code. Using a COND parameter, you can test the return code and, based on the test, either bypass or execute a step. The COND parameter can be specified on either a JOB or EXEC statement. See the *JCL User's Guide* for explanation and examples.

4.3.12 Comparison of VSE and MVS JCL - A Summary

Below is a summary of VSE JCL statements and possible equivalents in MVS.

Table 7 (Page 1 of 2). VSE Job Control Statements Summary

VSE Statement	Function	MVS Equivalent
ASSGN	Used at execution time to assign a specific device address to the symbolic unit name used.	// DD UNIT=
CLOSE	Closes either a system or a programmer logical unit assigned to tape, disk, or diskette.	No equivalent in OS/390. Units closed automatically at end-of-step, or by the application program.
DATE	Contains a date that is put in the communications region.	No equivalent in OS/390.
DLBL	Contains file label information for disk or diskette label checking and creation.	// DD DSName=
EXEC	Indicates the end of job control statements for a job step and that the job step is to be executed.	// EXEC PGM= (Must precede DD statements for the step.)
EXEC PROC	Calls a cataloged procedure and defines values for symbolic parameters.	// EXEC PROC= (″PROC=″ is optional)
EXTENT	Defines each area, or extent, of a disk file or diskette volume.	// DD VOLUME= SPACE= DATACLAS=
GOTO	Causes JCL to skip all following statements (except JOB, /&, /+) up to the specified label statement.	No direct equivalent in OS/390. Use CONDitional step processing or IF/THEN/ELSE/ENDIF
ID	Used to specify user identification and password.	// JOB USER= PASSWORD=
IF	Causes skipping or execution of the following statement dependent on the specified condition.	// IF
JCLEXIT	Activates or deactivates one or more JCL exit routines.	Not applicable in OS/390.
JOB	Indicates the beginning of control information for a job.	// JOB

LIBDEF	Defines library chains.	// STEPLIB DD,
		// STEPCAT DD, or
		// JCLLIB

LIBDROP	Drops library chain definitions.	No equivalent in OS/390.
LIBLIST	Lists library chain definitions.	No equivalent in OS/390
LIBSERV	Controls 3494 tape system.	No equivalent in OS/390
LISTIO	Used to get a listing of I/O assignments on SYSLOG or SYSLST.	No equivalent in OS/390
MTC	Controls operations on magnetic tapes.	// DD LABEL

Table 7 (Page 2 of 2). VSE Job Control Statements Summary

VSE Statement	Function	MVS Equivalent
ON	Causes specified action to be done if the specified condition is true after any step in the following job stream.	// EXEC COND= or //IF/ENDIF
OPTION	Sets one or more of the job control options.	// EXEC PARM=
PAUSE	Causes a pause immediately after processing this statement, or at the end of the current job step.	No equivalent in OS/390.
PROC	Defines and initializes symbolic parameters in a procedure.	// PROC
PWR	Passes a PRELEASE or PHOLD command to POWER.	/*$command
QUERY	Displays information on data spaces and standard options.	No equivalent in OS/390.
RESET	Resets I/O assignments to the standard assignments.	No equivalent in OS/390.
RSTRT	Restarts a checkpointed program.	// JOB RESTART=
SETPARM	Assigns a character string or return code to the specified parameter.	// SET
SETPFIX	Defines limits for PFIXing pages.	No equivalent in OS/390.
SETPRT	Loads the IBM 3800 buffers.	// DD or // OUTPUT
STDOPT	Resets system defaults.	No equivalent in OS/390.
SYSDEF	Defines limits and defaults for data spaces.	No equivalent in OS/390.
TLBL	Contains file label information for tape label checking and writing.	// DD dsname= , LABEL=
UPSI	(User Program Switch Indicators) Allows the user to set program switches that can be tested.	No equivalent in OS/390. (Use // EXEC PARM=)
VDISK	Defines the layout of a virtual disk.	No equivalent in OS/390.
ZONE	Initializes the zone field in the communications region.	No equivalent in OS/390.
/.	Label statement.	No equivalent in OS/390.
/*	Indicates the end of a data file.	/*
/&	Indicates the end of a job.	//
*	Job control comments.	See JES2 control statements below
/ +	Indicates the end of a procedure or librarian End-of-Data.	// PEND

4.3.13 Summary of MVS JCL Statements

Table 8. MVS Job Control Statements

JCL Statement	Purpose
// command	Enters an MVS system operator command through the input stream. The command statement is used primarily by the operator. Use the COMMAND statement instead of the JCL command statement.
// COMMAND	Specifies an MVS or JES command that the system issues when the JCL is converted. Use the COMMAND statement instead of the JCL command statement.
//* comment	Contains comments. The comment statement is used primarily to document a program and its resource requirements.
// CNTL	Marks the beginning of one or more program control statements.
// DD	Identifies and describes a data set. Indicates the end of data placed in the input stream. Note: Any two characters can be designated by the user to be the delimiter.
// ENDCNTL	Marks the end of one or more program control statements.
// EXEC	Marks the beginning of a job step; assigns a name to the step; identifies the program or the cataloged or in-stream procedure to be executed in this step.
// IF/THEN/ELSE/ENDIF	Specifies conditional execution of job steps within a job.
// INCLUDE	Identifies a member of a partitioned data set (PDS) or partitioned data set extended (PDSE) that contains JCL statements to be included in the job stream.
// JCLLIB	Identifies the libraries that the system will search for INCLUDE groups and Procedures named in EXEC statements.
// JOB	Marks the beginning of a job; assigns a name to the job.
//	Marks the end of a job.
// OUTPUT	Specifies the processing options that the job entry subsystem is to use for printing a SYSOUT data set.
// PEND	Marks the end of an in-stream or cataloged procedure.
// PROC	Marks the beginning of an in-stream procedure and may mark the beginning of a cataloged procedure; assigns default values to parameters defined in the procedure.
// SET	Defines and assigns initial values to symbolic parameters used when processing JCL statements. Changes or nullifies the values assigned to symbolic parameters.

4.4 JECL

JECL is very important in VSE and is commonly used. The difficulty from a conversion standpoint is to determine where the job is due to having two different job cards in JCL. The JECL statement is a POWER job, the DOS Job or VSE Job is the // JOB. The POWER job is like the MVS JOB. This is where the class and priority information is specified. It exists at the beginning of a job stream.

JES control statements are not recommended for new applications. You should use the new JCL statements such as // OUTPUT instead of the /*OUTPUT, /*ROUTE JECL statements. Today most JECL functions can be accomplished through standard JCL statements.

See Table 10 on page 90 for a list of recommendations.

4.4.1.1 LIST Card - * $$ LST

The LIST card in VSE is the equivalent of both // DD SYSOUT statement and OUTPUT statement in MVS.

Defaults, list and punch destinations can be put on the VSE JOB card as well as the LST card. Both are merged into the OUTPUT statement in MVS.

There is a difference between the scope of a LST statement and an equivalent DD statement. In MVS a DD statement is only viable for one step. In VSE, a * $$ LST statement is in effect from the time it is processed by POWER until EOJ or another LST statement for the same device address is processed. As soon as the effect of a given LST statement ends then the output is available for printing.

4.4.1.2 Data Statement - * $$ DATA

The data statement in VSE is a way for an include in SLI to specify the point where some data external to that include statement should be processed. It is similar to sending overrides in JECL but not as cumbersome in the space with the line number. The data statement allows you to pass data to a step. The step could be in the middle of an include statement. This is a commonly used method. The MVS equivalent is either a DD DATA override or a DD that points to a PDS member or a sequential data set.

4.4.2 Comparison of POWER and JES2 JECL - A Summary

Table 9 (Page 1 of 2). Overview of POWER JECL Statements

POWER Statement	Function	JES2 or MVS Equivalent
* $$ CTL	Assigns a new default input class to VSE/POWER jobs.	$T RDR(nn),Class=class
* $$ DATA	Inserts data from the reader queue into a library member after this member was retrieved for inclusion into a VSE/POWER job.	// DD * or // DD DATA
* $$ EOJ	Indicates the end of a VSE/POWER job.	//

Table 9 (Page 2 of 2). Overview of POWER JECL Statements		
POWER Statement	**Function**	**JES2 or MVS Equivalent**
* $$ FLS	Indicates that a VSE/POWER job should be terminated by internal flushing.	/*PURGE if INTRDR
* $$ JOB	Indicates the beginning of a VSE/POWER job and specifies the routing of jobs, output, and notify messages.	// JOB
* $$ LST	Provides handling information for printer output; routes list output to a node.	// OUTPUT, /*OUTPUT, or // DD SYSOUT=x, DEST=destination
* $$ PUN	Provides handling information for punched output; routes punch output to a node.	// OUTPUT, /*OUTPUT, or // DD SYSOUT=x, DEST=destination
* $$ RDR	Inserts a diskette file into the input stream.	No equivalent in OS/390.
* $$ SLI	Inserts data from an accessible library.	// INCLUDE
* $$ /*	Indicates the end of a VSE job step (used with the SLI statement only).	No equivalent in OS/390.
* $$ /&	Indicates the end of a VSE job (used with the SLI statement only).	No equivalent in OS/390.
/*$SLI	Indicates end of input data for an SLI member.	No equivalent in OS/390.

4.4.3 Summary of JES2 JECL - A Table

Table 10 (Page 1 of 2). JES2 Control Statements		
Statement	**Purpose**	**Comments**
/*$command	Enters JES2 operator commands through the input stream.	
/*JOBPARM	Specifies certain job-related parameters at input time.	Use parameters on the // JOB statement instead.
/*MESSAGE	Sends messages to the operator via the operator console.	Seldom used.
/*NETACCT	Specifies an account number for a network job.	Seldom used.
/*NOTIFY	Specifies the destination of notification messages.	Use NOTIFY on the // JOB or // OUTPUT statements.
/*OUTPUT	Specifies processing options for SYSOUT data set(s).	Use the // OUTPUT JCL statement instead.
/*PRIORITY	Assigns a job queue selection priority.	Use PRTY= on the // JOB statement instead.
/*ROUTE XEQ	Specifies the execution node for the job.	Use the /*XMIT statement as an alternative.

Table 10 (Page 2 of 2). JES2 Control Statements		
Statement	**Purpose**	**Comments**
/*ROUTE PRT or /*ROUTE PUN	Specifies the default print or punch destination for the job.	Use the // OUTPUT DEFAULT=YES instead.
/*SETUP	Requests mounting of volumes needed for the job.	Seldom used. (Similar to VSE PAUSE statement)
/*SIGNOFF	Ends a remote job stream processing session.	BSC RJE Workstation use only.
/*SIGNON	Begins a remote job stream processing session.	BSC RJE Workstation use only.
/*XEQ	Specifies the execution node for a job.	(Short form of /*ROUTE XEQ)
/*XMIT	Indicates a job or data stream to be transmitted to another NJE node.	A // JOB card must precede this, and a job statement for the execution node must follow this.

4.5 VSE and MVS JCL Comparison Example

The following example jobs (4.5.1, Sample VSE JCL, 4.5.2, Sample MVS JCL, and 4.5.3, Sample VSE plus Carry-Over) show different ways to code the JCL to execute PROGRAM1, SORT, and PROGRAM2. Though these jobs appear to be different, the output is exactly the same in each example.

- Step (Job) 1

  PROGRAM1 reads data from TAPEIN (INPUT-TAPE in VSE and INPUT.TAPE in OS/390) and writes data to DISKOUT (WORK-DISK in VSE and WORK.DISK in OS/390).

- Step (Job) 2

  SORT takes data from SORTIN (WORK-DISK in VSE and WORK.DISK in OS/390) and writes sorted data to SORTOUT (WORK-DISK 2 in VSE and WORK.DISK2 in OS/390).

- Step (Job) 3

  PROGRAM2 reads data from DISKIN (WORK-DISK 2 in VSE and WORK.DISK2 in OS/390) and sends output to two different locations (Endicott and Boeblingen).

By comparing the file definitions described above you can see which JCL statements in VSE and MVS perform equivalent functions (TLBL or DLBL/EXTENT equate to DD, EXEC equates to EXEC, and so on). Notice also the very slight difference in syntax: VSE has a space after the ′//′, MVS does not unless it is a continuation card, also VSE continuation starts on column 16. In VSE, the file definitions precede the EXEC statement while in MVS they succeed the EXEC statement.

The JCL in ″4.5.1, Sample VSE JCL″ and ″4.5.2, Sample MVS JCL ″ show an equivalent relationship as to the placement of file definitions in the different steps (that is, the file definitions are all in the step where they are used). By contrast ″4.5.3, Sample VSE plus Carry-Over″ shows how file definitions can all be located at the beginning of the VSE JCL and ″carried″ throughout the entire

job (all definitions available to all steps). OS/390 operation does not perform this ″carry-over″ (unique to VSE).

4.5.1 Sample VSE JCL

This example shows one POWER job containing three VSE jobs.

```
* $$ JOB JNM=MYJOB,CLASS=F,USER='ITSO SAMPLE'
* $$ LST LST=SYSLST,JSEP=0,CLASS=W,COPIES=3
// JOB JOB1              extract records from tape
// ASSGN SYS005,480          input tape
// TLBL TAPEIN,'INPUT-TAPE'
// DLBL DISKOUT,'WORK-DISK',0,SD
// EXTENT DISK01,1,0,100,500
// EXEC PROGRAM1,SIZE=AUTO
// MTC SYS005,RUN           unload tape
/*
*        check previous job
// PAUSE in case it abended
/*
/&
// JOB JOB2     SORT WORK FILE BY PLANT NUMBER
* $$ LST LST=SYSLST,JSEP=0,CLASS=A
// DLBL SORTIN,'WORK-DISK',0,SD
// EXTENT DISK01,0
// DLBL SORTOUT,'WORK-DISK 2',0,SD
// EXTENT DISK14,0,600,500
/ DLBL SORTWK1,'%%SORT.WORK1',0,VSAM,RECSIZE=100,RECORDS=50000,        C
               DISP=(NEW,DELETE)
// DLBL SORTWK2,'%%SORT.WORK2',0,VSAM,RECSIZE=100,RECORDS=50000,       C
               DISP=(NEW,DELETE)
// EXEC SORT,SIZE=200K
 SORT FIELDS=(1,32,CH,A),WORK=2
 RECORD TYPE=F,LENGTH=87
 INPFIL BLKSIZE=4350
 OUTFIL BLKSIZE=4350
/*
*     SORT ENDED
/&
// JOB JOB3     PRINT REPORT
// DLBL DISKIN,'WORK-DISK 2',0,SD
// EXTENT DISK14,0
// DLBL PRODCAT,'PROD.USER.CATALOG',,VSAM
// DLBL MASTER,'PLANT.MASTER.FILE',,VSAM,CAT=PRODCAT
// EXEC PROGRAM2,SIZE=300K
* $$ LST LST=SYS010,DEST=KCJONES
01 ENDICOTT
* $$ LST LST=SYS010,DEST=HERBERT
02 BOEBLINGEN
/*
/&
* $$ EOJ
```

4.5.2 Sample MVS JCL

The task surrounding the conversion of JCL is more than mapping between VSE JCL using this syntax and MVS JCL using that syntax. At the base of these two JCLs are different philosophies. A parameter by parameter comparison is insufficient. Comparing the VSE DLBL/EXTENT to a DD Statement is only part of the story. These examples are meant to give the read only a ″flavor″ for what changes have to take place. It is necessary to look at the two systems at a higher level as well.

```
//MYJOB JOB ACCT#,'REPORT BY PLANT',CLASS=F,REGION=4M
//*
//STEP1 EXEC PGM=PROGRAM1
//SYSLST DD SYSOUT=W COPIES=3
//TAPEIN DD DSN=INPUT.TAPE,DISP=OLD,
// UNIT=TAPE,VOL=SER=REEL22
//DISKOUT DD DSN=WORK.DISK,DISP=(,CATLG),
// UNIT=SYSDA,SPACE=(TRK,(500,100),RLSE)
//*
//STEP2 EXEC PGM=SORT,COND=(0,NE)
//SORTIN DD DSN=WORK.DISK,DISP=(OLD,DELETE)
//SORTOUT DD DSN=WORK.DISK2,DISP=(,CATLG),
// UNIT=SYSDA,SPACE=(TRK,(500,100),RLSE)
//SYSOUT DD SYSOUT=*
//SYSIN DD *
 SORT FIELDS=(1,32,CH,A)
 RECORD TYPE=F,LENGTH=87
/*
//SORTWK01 DD UNIT=SYSDA,SPACE=(TRK,(500,100))
//SORTWK02 DD UNIT=SYSDA,SPACE=(TRK,(500,100))
//*
//STEP3 EXEC PGM=PROGRAM2,COND=(0,NE)
//SYSLST DD SYSOUT=F,FCB=FK33
//DISKIN DD DSN=WORK.DISK2,DISP=(OLD,DELETE)
//DEST01 DD SYSOUT=A,DEST=KCJONES
//DEST02 DD SYSOUT=A,DEST=HERBERT
//SYSIN DD *
01 ENDICOTT
02 BOEBLINGEN
```

1. Conditional OPENs

   An example of the higher level differences between OS/390 and VSE is in the area of allocation. In OS/390, allocation is at the beginning of the step. VSE does not open a file until an OPEN is issued. This is a key concept and one that requires more than a term by term comparison. Frequently in VSE there are applications that open a file for output once a week on Friday. On other days this VSE file won't be opened. In OS/390, a data set will be created every day whether it is used or not. These high level type differences must also be addressed and as early in the migration as possible.

2. In-stream DD Card

   There are three or four techniques to handle these situations.

   One method is to convert the imbedded in-stream DD CARD by having the JCL Batch of DD Names that somehow tie up the control cards and have PROGRAM2 call some subroutine that would change the DD Name.

A method that used two SYSOUT devices and output two DCBs in the program could also work.

3. PROGRAM2 Changes

   PROGRAM2 has to change for OS/390 to simulate the imbedded LST cards in the VSE job (for DEST=KCJONES and DEST=HERBERT). PROGRAM1 was OK as far as file assignments.

4.5.3 Sample VSE plus Carry-Over

In this example the job cards for JOB2 and JOB3 have been commented out making this a POWER job that contains one VSE job with three jobsteps. Also, the file definitions have all been moved to the beginning of the job. This demonstrates the ″carry-over″ effect where the file definitions are available (″carry-over″) to all steps within the VSE job.

```
* $$ JOB JNM=MYJOB,CLASS=F,USER='ITSO SAMPLE'
* $$ LST LST=SYSLST,JSEP=0,CLASS=W,COPIES=3
// JOB JOB1               EXTRACT RECORDS FROM TAPE
// ASSIGN SYS005,480          INPUT TAPE
// TLBL TAPEIN,'INPUT-TAPE'
// DLBL DISKOUT,'WORK-DISK',0,SD
// EXTENT DISK01,0,100,500
// DLBL SORTIN,'WORK-DISK',0,SD
// EXTENT DISK01,0
// DLBL SORTOUT,'WORK-DISK 2',0,SD
// EXTENT DISK14,0,600,500
// DLBL SORTWK1,'%%SORT.WORK1',0,VSAM,RECSIZE=100,RECORDS=50000,       C
               DISP=(NEW,DELETE)
// DLBL SORTWK2,'%%SORT.WORK2',0,VSAM,RECSIZE=100,RECORDS=50000,       C
               DISP=(NEW,DELETE)
// DLBL DISKIN,'WORK-DISK 2',0,SD
// EXTENT DISK14,0
// DLBL PRODCAT,'PROD.USER.CATALOG',,VSAM
// EXTENT PROD22,0
// DLBL MASTER,'PLANT.MASTER.FILE',,VSAM,CAT=PRODCAT
/*
// EXEC PROGRAM1,SIZE=AUTO
// MTC SYS005,RUN           UNLOAD TAPE
/*
*        CHECK PREVIOUS JOB
// PAUSE IN CASE IT ABENDED
/*
** JOB JOB2     SORT WORK FILE BY PLANT NUMBER
* $$ LST LST=SYSLST,JSEP=0,CLASS=A
// EXEC SORT,SIZE=200K
 SORT FIELDS=(1,32,CH,A),WORK=2
 RECORD TYPE=F,LENGTH=87
 INPFIL BLKSIZE=4350
 OUTFIL BLKSIZE=4350
/*
*     SORT ENDED
```

```
** JOB JOB3     PRINT REPORT
// EXEC PROGRAM2,SIZE=300K
* $$ LST LST=SYS010,DEST=KCJONES
01 ENDICOTT
* $$ LST LST=SYS010,DEST=HERBERT
02 BOEBLINGEN
/*
/&
* $$ EOJ
```

Chapter 5. Disk and Tape Storage Considerations

The VSE/SP and VSE/ESA systems and MVS and OS/390 systems have some conceptual similarities and data compatibilities for disk and tape files. This chapter will discuss the similarities and differences between the VSE and OS/390 environments in the following areas:

- 5.1, Access Method Similarities and Differences
- 5.2, Data Set Naming Considerations
- 5.3, Storage and Space Management
- 5.4, Tape Similarities and Differences
- 5.5, DASD Similarities and Differences
- 5.6, VSAM Differences

These topics will be discussed in order within this chapter.

5.1 Access Method Similarities and Differences

5.1.1 Access Methods

An access method is a set of user application programming interfaces (APIs), utility programs, other programming, and format standards which provide users with the ability to readily store and retrieve data within a computer system.

The VSE/ESA and OS/390 operating systems each support a number of access methods, with varying levels of compatibility. Further, details of the implementation are different between the operating systems, even if the function and the external format of the data are the same.

Access methods used for disk and tape storage in the VSE system include the following:

SAM Sequential Access Method -- used for disk and tape devices. Records are stored and/or retrieved in the order presented.

In OS/390, the most similar access methods are QSAM or SAM-E.

ISAM Indexed Sequential Access Method -- formerly used for disk devices, when records were to be maintained (logically) in ascending key sequence, but might need to be retrieved in arbitrary order (by key). Obsolete, replaced by VSAM in most VSE environments over twenty years ago.

In OS/390, may still be supported, but not recommended.

VSAM Virtual Storage Access Method -- used for disk devices. Records can be stored and/or retrieved in the order presented, or in key or address order.

In OS/390, DFP/VSAM is the most similar access method.

VSAM will be discussed in a separate sub-chapter below.

DAM (or BDAM)

Direct Access Method (or Basic Direct Access Method) -- used for disk devices. Still in some use, but often replaced by VSAM functions in many VSE shops. Generally requires complex application handling for processing, and may be dependent upon physical device characteristics. Not supported on Fixed-Block Architecture disks.

In OS/390, BDAM is the functional equivalent.

Libraries VSE Librarian should be considered an access method, as it meets all the criteria specified above. The current VSE Librarian has been available since VSE/SP Version 2, in 1984. The previous implementation will not be discussed.

In OS/390, Partitioned Data Sets (PDS, PDS-E) provide the equivalent functions, together with associated utilities.

5.1.2 Operating System Implementations

In the VSE/ESA system, programs define their intent to use an access method and specify needed parameters through the APIs provided through a set of Define The File macros. These include:

DTFCD Define The File CarD

DTFCN Define The File CoNsole

DTFDA Define The File Direct Access

DTFDI Define The File Device Independence

DTFDR Define The File Document Reader

DTFDU Define The File Diskette Unit

DTFIS Define The File Indexed Sequential access

DTFMR Define The File Magnetic ink character Reader

DTFMT Define The File Magnetic Tape

DTFOR Define The File Optical Reader

DTFPH Define The File for PHysical I/O

DTFPR Define The File for PRinter

DTFSD Define The File for Sequential Disk

In addition to these, additional macros are available for definition of VSAM, Librarian, and some other access method objects or files, including telecommunication terminals or lines.

In the OS/390 environment, most of these DTFs are replaced by an analogous control block definition, the Data Control Block, or DCB. The DCB is not device specific, as the VSE DTFs are, so there is more flexibility for using a single OS/390 program to read data, for example, from the SYSIN stream, from tape or from disk. Only the JCL would be changed to specify the device characteristics at run time to switch from one input or output device type to another.

The VSE application program contains the DTF macro expansion, and at linkage edit or execution time, an operating system component module (referred to as a ″Logic Module″) will be connected to the DTF and used by the application program to handle the functions needed for GET, PUT, or other imperative macro commands.

In OS/390, the application program linkage is handled through the SVC interfaces of the operating system.

In either case, the application program functional request (GET, PUT, and so on) will cause the next logical record to be retrieved from a buffer or from external media. Channel programs are created, and the operating system will cause the channel programs to be scheduled to perform any physical I/O operations required by the functional request.

5.1.3 Miscellaneous Functions

In VSE and OS/390 environments, access method OPEN logic is responsible to ensure that the user is authorized to access the data in the file being OPENed. This includes label (Data Set Control Block or DSCB) checking for input files, and checking to ensure that the output file does not overlay existing files. In addition, the access method (and operating system) is responsible for error handling and recovery where possible only notifying the application program when unrecoverable errors occur.

In OS/390, the access method will interface with the operating system Direct Access Device Space Management (DADSM) component to manage allocation of space as required. In VSE, this function may be done through VSE/VSAM Space Management, an OEM vendor product, or manually through JCL specifications.

5.2 Data Set Naming Considerations

5.2.1 VSE Considerations

It is common in VSE shops to have loose data set naming standards. System files may be named in a standard fashion, but application files will be named depending on the programmer or implementer of the application. Identifying files by application or subsystem from their name may be difficult, and knowledge of how one installation has named their files will be of little use in another installation.

For non-VSAM files, the format of file-ids (between the apostrophe characters) is not defined by the system. One continuous string of up to 44 characters may be defined as the file-id for a disk file using the VSE DLBL JCL statement.

In most VSE shops, this will provide little problem -- a rare case of aggravation, at most, and then only rarely. This is not the case in OS/390 environments, however.

5.2.2 OS/390 Considerations

In OS/390 environments, the connection identifying which user catalog contains the management information for a given file or data set is dependent upon OS/390′s ALIAS mechanism. Further, the specific requirement for at most eight characters between periods in any data set name (DSN) is enforced. Each string of characters (node) must start with an alphabetic or national character, and the system uses the high order (left-most) node to identify which user catalog contains the catalog information for the data set in question. The system′s MASTER catalog contains the list of ALIAS definitions together with the user catalog associations.

As it is desirable from both performance and integrity perspectives to separate user data sets into several user catalogs, it is critical that data set names be defined with the above information in mind. If, for example, all data set names begin with CUSTOMER as the first node, then all the data sets will be defined in only one user catalog. If the high order node for some data sets is PAYROLL, and for others INVENTRY, then these data sets could be split between two user catalogs if desired.

It is strongly recommended that data set naming conventions be defined carefully and that they be enforced through training and review procedures during conversion and after the system has been placed into production.

As data set naming conventions can greatly simplify the operation of your OS/390 system, and can also make the migration process simpler, we have included a separate chapter on recommendations for data set naming. See the chapter, 25.4, “Set Up Standards, Procedures, and Documentation” on page 407, for more information.

5.3 Storage and Space Management

5.3.1 VSE Considerations

Standard VSE system facilities provide for user management of DASD space resources, or for VSAM management of DASD space for SAM files. Many VSE users have OEM vendor disk space management packages, and/or tape library management packages. User management requires manual procedures to maintain catalogs of disk space in use or available, as well as tape volumes in use, and the relationships between tape files and tape volumes. VSE users who have not automated some or all of these functions find manual procedures error prone and slow.

5.3.2 OS/390 Considerations

OS/390′s Direct Access Device Storage Manager (DADSM) and Data Facility -- System Managed Storage (DFSMS) components provide all the capabilities of the VSE system, plus OEM vendor enhancements.

5.3.3 System Managed Storage

System-managed storage is the OS/390 automated approach to managing your system′s storage resources. It uses software programs to manage data security, placement, migration, backup, recall, recovery, and deletion to ensure that current data is available when needed, and obsolete data is removed from storage. The combination of system-managed storage and related hardware and software products is called the DFSMS environment.

With DFSMS, you tailor the system-managed storage environment to your needs. You define the requirements for performance, security, and availability, along with storage management policies used to automatically manage the direct access, tape, and optical devices used by the OS/390 operating systems. The Interactive Storage Management Facility (ISMF), a component of DFSMSdfp, provides the user interface for defining and maintaining these policies and the Storage Management Subsystem (SMS) governs these policies for the system. In this environment, RACF and DFSORT compliment the functions of the base operating system. The following are brief descriptions of the DFSMS/MVS

functional components as well as RACF and DFSORT. For details, see section 1.3 in the *DFSMS/MVS General Information*, GC26-4900.

- **DFSMSdfp** is an OS/390 base element and functional component of DFSMS/MVS. It provides the storage, program, data, and device management functions of MVS. DFSMSdfp provides the foundation for distributed data access, using the Distributed File Manager to support remote access of MVS data and storage resources from workstations, personal computers, or other authorized systems in an SNA LU 6.2 network. DFSMSdfp also provides the Storage Management Subsystem component. DFSMSdfp is the DFSMS component that provides OS/390′s VSAM functions.
- **DFSMSdss** is an OS/390 optional feature and functional component of DFSMS/MVS. It is used for data movement and replication, space management (defragmentation), data backup and recovery, and data set and volume conversion to system managed storage.
- **DFSMShsm** is an OS/390 optional feature and functional component of DFSMS/MVS. It provides the functions for:
  - **Storage management** using a hierarchy of storage devices in its automatic management of data, relieving end-users from manual storage tasks.
  - **Space management** by keeping only active data on fast-access storage devices. It automatically frees space on user volumes by deleting eligible data sets, releasing over-allocated space, and moving low-activity data to lower-cost-per-byte devices.
  - **Tape mount management** by writing multiple output data sets to a single tape, making it a useful tool for implementing tape mount management (TMM) under SMS. For more information on TMM, refer to *Implementing System-Managed Storage*, SC26-3123.
  - **Availability management** by backing up your data, either automatically or by command, to ensure availability in the event of accidental loss of data sets or physical loss of volumes. Disaster backup and recovery is also provided for user-defined groups of data sets (aggregates), so that critical applications can be restored at the same location or at an off-site location. This capability is referred to as ABARS.
- **DFSMSrmm** is an OS/390 optional feature and functional component of DFSMS/MVS. It provides the management functions for removable media, including tape cartridges and reels. DFSMSrmm can be used to manage system-managed tape libraries such as the 3494 or 3495 Automated Tape Library Dataservers or the manual 3495 Tape Library Dataserver Model M10. DFSMSrmm can also manage non-system-managed tape libraries.
- **DFSORT** is an OS/390 optional feature. It sorts, merges and copies data sets, and also helps you to analyze data and produce detailed reports using the ICETOOL utility or the OUTFIL function.
- **RACF** is an OS/390 optional feature. It controls access to data and other resources in MVS.

For more information on the benefits of system-managed storage, refer to the following publications:

DFSMS/MVS General Information, GC26-4900
Implementing System-Managed Storage, SC26-3123
DFSMS/MVS Planning for Installation, SC26-4919
RACF General Information, GC23-3723
Getting Started with DFSORT, SC26-4109

5.3.4 Implementing DFSMS

Implementation of DFSMS is not required for OS/390, but many of the newer functions available with OS/390 and DFSMS/MVS require system-managed data and the associated activation of the Storage Management Subsystem (SMS) address space. The following is a small sampling of some of the functions requiring data that is system-managed:

- Allocation of OpenEdition Hierarchical File System (HFS) data sets.
- Allocation of Extended Format data sets which provides:
  - VSAM System-Managed-Buffering (SMB)
  - VSAM Extended Addressability
  - VSAM Compression
  - SAM Tailored Compression
- Reduction of out-of-space failures
- VSAM Record Level Sharing (RLS)

A complete list of new functions available with DFSMS/MVS and considerations for implementing them can be found in *DFSMS/MVS Planning for Installation*, SC26-4919.

DFSMS implementation can be a phased migration using a milestone approach. The five major milestones are:

1. Enabling the software base
2. Activating the storage management subsystem
3. Managing temporary data
4. Managing permanent data
5. Managing tape data

Using the milestone approach allows you to begin with low-risk implementation activities that establish a base for the staged migration of your data to storage management. In later milestones, your early experience is used to achieve greater data and storage automation. The milestone process is documented in *Implementing System-Managed Storage*, SC26-3123. Please refer to the publication for more detail.

In conjunction with the milestone approach to implement DFSMS, you can use the DFSMS Fast Implementation Technique (FIT) to plan the implementation. DFSMS FIT uses a question-and-answer approach to create a DFSMS design tailored to your installation′s needs, and a data classification system that allows you to use your data set naming standards already in place. It helps you quickly identify the different types of data to which you want to assign specific data-set-level, SMS-management policies.

DFSMS FIT is documented in the following IBM International Technical Support Organization publications (Redbooks):

Get DFSMS FIT: Fast Implementation Techniques, SG24-2568
DFSMS FIT: Fast Implementation Techniques Process Guide, SG24-4478
DFSMS FIT: Fast Implementation Techniques Installation Examples, SG24-2569

In conjunction with DFSMS FIT, you can use NaviQuest. NaviQuest is a component of DFSMSdfp and is an option from the ISMF panels. With NaviQuest you can:

- Automatically test your DFSMS configuration
- Automatically create test cases
- Automatically test your ACS routines
- Perform storage reporting, through ISMF and with DCOLLECT and VMA data
- Print ISMF lists
- Run ISMF functions in batch mode, using the REXX EXECs provided

For more information on NaviQuest, please refer to the *NaviQuest User's Guide*, SC26-7194.

There are DFSMS education courses available to learn more about DFSMS and how it can be implemented. To find information concerning IBM Education and Training's storage systems curricula for your area, contact your IBM Representative or the IBM E&T Web site at:
http://www.training.ibm.com/ibmedu/roadmaps/mainframe/storsys/

In addition, many users have found the ADSTAR Distributed Storage Manager (ADSM) to be of great value when LAN attached workstations are part of your overall system solution.

DFSMS is a standard part of your OS/390 system. Use its facilities to ease your migration and subsequent operations.

5.4 Tape Similarities and Differences

5.4.1 Volume Interchangeability

Tape label conventions, requirements, and handling techniques differ between VSE and OS/390 systems. However, OS/390 should be able to read and process all tapes that have been created on a VSE system. Similarly, VSE systems should be able to read tape volumes created by OS/390 systems. The rest of this chapter explains tape label differences and migration considerations.

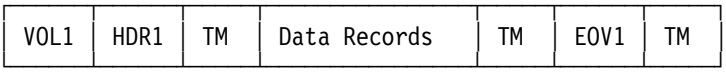
5.4.2 Standard Labels

Tapes with standard labels written by VSE can be read by OS/390. Standard labels are 80-character records recorded in EBCDIC and odd parity on 9-track tape, BCD and even parity with translation, on 7-track tape. The first four characters always identify the labels.

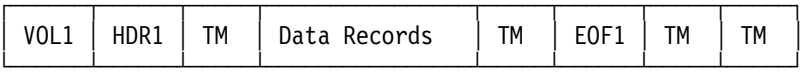
VOL1	volume label
HDR1 and HDR2	data set header label
EOV1 and EOV2	data set trailer labels (end of volume)
EOF1 and EOF2	data set trailer labels (end of data set)
UHL1 - UHL8	user header labels
UTL1 - UTL8	user trailer labels

Note: UHL and UTL processing are user standard labels.

Under VSE, standard label write processing when FILABL=STD is specified in the DTF includes:



for the first and intermediate volumes of a multivolume file, and



for the last volume of a multivolume file.

In addition to VSE formats, OS/390 standard labels, identified by the DD parameter LABEL=(,SL), include HDR2, EOV2 and EOF2 labels. If these labels are on input tapes under VSE, they are skipped; on output tapes, they are overwritten. OS/390 accepts the absence of these labels when processing input data sets.

OS/390 obtains information about the characteristics of a tape file (for example, blocksize) from three different sources:

1. The tape label (for labelled tapes)
2. The DCB parameter of the DD job control statement (JCL) that identifies the data set
3. The DCB macro specification (assembler) or comparable high-level language file specification within the source program. (Coding these in a program is **not recommended**; that is, recoding and relinking of the program is required when device type, blocksizes or other data set changes are desired.)

For output tapes, specifying the file characteristics via JCL specifications is recommended. It provides flexibility by allowing OS/390 to make final device (in this case, tape drive) assignments at job execution time.

VSE file-ids can be up to 17 characters in length, and do not have to be qualified; that is, the 17 characters do not have to have any periods in the string of characters. In OS/390, file names (data set names) must have qualifiers if more than eight characters are used in the name (that is, a period must be used in the file name with no more than eight characters in a qualifier). **If VSE created file-ids are longer than eight characters and unqualified, the files can be read in OS/390 only by specifying their file-id in apostrophes**. This is done in the JCL DSN= parameter; for example, DSN=′payroll/datafile52134′.

If HDR2 tape labels are not present on data sets used as input to OS/390 (HDR2s aren′t created by VSE systems), you must supply the

- block length
- record length
- tape recording technique (seven-track only)
- tape density
- record format

to the OS/390 system. This information was coded in VSE programs. However, it is strongly recommended that this data be removed from VSE programs as they are being converted to OS/390 versions. Place the above data in the DCB subparameter of the DD JCL statement specifying the tape file.

5.4.2.1 Standard User Labels

Differences between VSE and OS/390 in the area of standard user labels exist at the application program level.

With VSE you must supply a routine to check or build standard user labels. Specify the symbolic address of your label routine in the DTFMT, DTFSR, or DTFPH entry in LABADDR=name. The storage address of this routine is available in register 15. The IOCS OPEN and CLOSE routines branch to your label routine after processing the standard labels for a file. At the end of the routine, return to IOCS by issuing an LBRET instruction.

Under OS/390, you can specify the address of the standard user label processing routines through the EXLST (exit list) parameter of the DCB associated with the tape data set. An additional specification is made in the job control DD statement.

Code the LABEL parameter of the DD statement associated with the data set as LABEL=(,SUL) if the tape contains (or the tape is to be created with) both standard and user labels. If either SUL or EXLST is omitted, user label processing is bypassed. As under VSE, user labels are processed during OPEN and CLOSE.

Processing of standard user labels on input tapes is skipped if you don't specify the name of a user routine in the DTF or DCB. Standard user labels are optional on output tapes and are created only if defined in your label exit logic.

5.4.3 No Labels

An unlabeled tape contains only data records and tapemarks. Unlabeled tapes created by VSE generally have a tapemark preceding the first file; OS/390-created unlabeled tapes do not. See Figure 7 on page 107 for a representation of VSE and OS/390 unlabeled tapes.

Unlabeled output tapes produced by VSE assembler language programs do not have a leading tapemark before the first file if the parameters TPMARK=NO and FILLABL=NO are included in the DTFMT used to create the tape. Unlabeled output tapes produced by VSE PL/I programs do not have a leading tapemark before the first file if NOTAPEMARK and NOLABEL are included in the ENVIRONMENT options list of the VSE PL/I file declaration. All output tapes produced via VSE programs written in other source languages *do* have a tapemark in the first record.

On OS/390 systems, to position a tape at the desired data set, you must specify the correct data set sequence number in the LABEL parameter of the DD

statement. If a tapemark might precede the first data set and you specify the LABEL subparameter LTM, OS/390 tests for and bypasses a leading tapemark, if present. If a tapemark precedes the first data set and you do not specify LTM in the LABEL parameter field, you must add one to the data set sequence number.

If a multivolume data set has a leading tapemark on one or more of the volumes, OS/390 can process it as an unlabeled multivolume data set if the LABEL subparameter LTM is specified. Otherwise, OS/390 cannot process it as an unlabeled multivolume data set.

The presence of a leading tapemark makes each data set the second in sequence on the tape. However, OS/390 always assumes that data sets are first in sequence on the tape. **Specify LTM in the LABEL parameter field, so the first data set on a tape can be accessed whether or not it is preceded by a leading tapemark**, for example code LABEL=(1,LTM).

The specification of LTM in the LABEL parameter field does not make allowances for any other tapemarks. You must make any such adjustments in the data set sequence number.

5.4.4 Nonstandard Labels

Nonstandard labels do not conform to standard label formats. They are designed by the installation, and are written and processed by routines provided by the installation. There are no requirements as to the length, format, contents, and number of nonstandard volume labels. When processing nonstandard labels, you must perform many of the functions that control program or IOCS performs in processing standard labels. All input/output operations, such as reading and writing labels, are performed by using the EXCP macro.

When nonstandard labels are to be created or checked, a user routine is required. Because of the basic differences between VSE and OS/390 in handling nonstandard labels, this portion of VSE programs requires rewriting.

Under VSE, nonstandard label processing is included as part of the user program when the DTFMT contains the parameters, FILABL=NSTD and LABADDR=name. IOCS OPEN and CLOSE routines provide the entry point for user processing of nonstandard labels according to the LABADDR parameter in the DTFIMT. If you omit this parameter with nonstandard labels, IOCS bypasses label processing. It is your responsibility to conform to standard register conventions. Only one nonstandard label routine is supported for each file, but multiple DTFs may share a common routine. No VSE job control language statement is required to define a nonstandard label. Figure 7 on page 107 illustrates nonstandard labels supported under VSE.

5.4.5 Bypass Label Processing Facility in OS/390

Should you run across VSE labels that cannot be processed by OS/390, you can use the OS/390 ″bypass label processing″ facility. Specifying BLP in the OS/390 JCL LABEL parameter (LABEL=(,BLP)) requests the system to bypass label processing. When used, the operator must ensure that the correct tape volume is mounted. **The BLP parameter should only be used for unique situations, and its use should be controlled.**

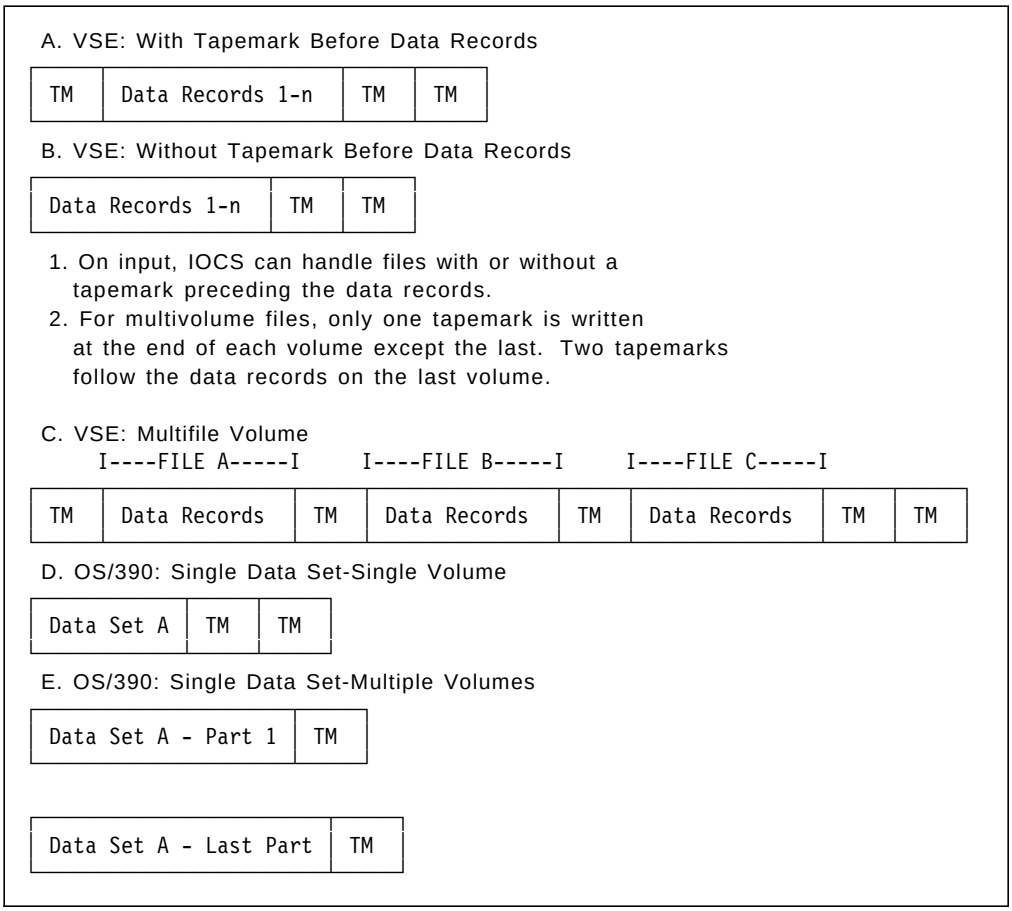


Figure 7. Nonstandard Labels Supported by VSE

When analyzing VSE programs supporting nonstandard labeled tape files used under OS/390, you must:

1. Create nonstandard label processing routines for input header labels, input trailer labels, output header labels and output trailer labels.
2. Insert the installation′s routines into the system by including predefined member names into SYS1.LPALIB as type4 SVC routines. (See *Using Magnetic Tapes*, GC26-4923.)
3. Code NSL in the job control LABEL parameter of the DD statement for the tape data set at execution time.

The first two items define the programming effort to be considered in the design and writing of the installation′s nonstandard label routines. Once these routines have been made a part of the SVC library, they are used by all application programs with tape data sets defined by LABEL=(,NSL).

The conventions, requirements, and techniques for writing OS/390 nonstandard label routines differ from the individual label processing subroutines used under VSE.

5.5 DASD Similarities and Differences

5.5.1 Volume Interchangeability

DASD file label conventions, requirements, and handling techniques differ between VSE and OS/390 systems. However, OS/390 should be able to read and process DASD volumes that have been created on a VSE system. Similarly, VSE systems should be able to read volumes created by OS/390 systems. The following exceptions to this are noted below.

- OS/390 doesn′t support FBA (Fixed Block Architecture) DASD devices; for example, 3310, 3370, 9332, 9335, 9336. Data files located on FBA devices will have to be copied to an OS/390-supported device using a VSE backup/restore program.
- Concurrent sharing of a volume between the two systems is **not supported** and should not be attempted as data loss could result.
- OS/390 does not allow more than one volume with an identical volume serial number (VOLID) to be online at any one time. VSE would allow this condition. Thus, if you are used to operating with identical serial number volumes (either tape or DASD) mounted concurrently, you will have to change to unique volume serial number identifiers for OS/390 operations.
- It is **not recommended** that OS/390 volumes which contain Indexed VTOCs be accessed by a VSE system. To do so requires special procedures - See 5.5.3, “Indexed VTOC Considerations (OS/390)” on page 109.

5.5.2 DASD (VTOC) Processing

DASD volumes are managed by both VSE and OS/390 through a Volume Table of Contents (VTOC) which is a type of file. There is always one VTOC per volume. Each VTOC is composed of various records called Labels in VSE, but named Data Set Control Blocks (DSCBs) in OS/390. DSCBs are used to store information about the contents of the volume. DSCBs come in various record formats with varying field layouts, and are customized by the type of information stored. There can be up to six different DSCB types in a VTOC. They are appropriately named: ″Format-1 DSCB ,..., Format-6 DSCB″. For a detailed description of a VTOC layout, index VTOC and the various DSCBs, see the *DFSMSdfp Advanced Services.* Also, see *VSE/AF Data Management Concepts*, GC33-6192. For purposes of this publication, only the Format-1, Format-4, and Format-5 DSCB need to be discussed.

Both VSE and OS/390 use the Format-1 DSCB. There is a Format-1 DSCB record for each data set allocated on a volume. OS/390 stores information about each data set in addition to the information which VSE stores. As such, VSE does not maintain all fields in the Format-1 DSCB. For example, the record format, record length, and block size are not maintained by VSE. Thus, to access VSE-created files from an OS/390 system, the user must supply (preferably through JCL) file information which otherwise would be obtained from the file′s Format-1 label or DSCB.

Both VSE and OS/390 use a Format-4 DSCB -- there is one (and only one) per VTOC. The Format-4 DSCB contains a 1-bit flag designator (called the ″DOS bit″) which is used to indicate whether the Format-5 DSCB(s) are valid. Turning the bit ″on″ or ″off″ is the responsibility of the operating system. VSE when it first uses a volume turns the ″DOS bit″ ″on″. The use of this bit is mainly for the purpose of

maintaining the accuracy of the Format-5 DSCB. ***The Format-5 DSCB***, sometimes called the ″free space DSCB″, ***is used only by OS/390***. OS/390 keeps track of unallocated space on a volume by creating one or more Format-5 DSCBs; that is, each Format-5 contains up to 26 extents of free space information.

VSE, although it doesn′t keep track of free space on a volume, will still be able to process a volume created by OS/390 (and on which a Format-5 DSCB is located). However, VSE does not update the Format-5 DSCB when new space is allocated, or allocated space is freed. Although VSE ignores Format-5 DSCBs, it ″tells″ OS/390 that the Format-5 DSCBs may no longer be valid (for example, a new file being allocated). This ″telling″ is accomplished through the ″DOS bit″ (previously discussed). OS/390, when it first allocates to a volume, always checks the Format-4 DSCB and the ″DOS bit.″ If it finds this bit ″on″, OS/390 creates (by invoking a VTOC, conversion routine) new Format-5 DSCBs representing all unallocated space it finds on the volume. Should there already be Format-5 DSCBs on a volume from prior OS/390 uses, OS/390 invalidates these and creates new ones. (An appropriate OS/390 message is displayed on the operator console whenever new Format-5 DSCBs are created as a result of VSE′s previous use of the volume.)

You may obtain OS/390 dump, formatted, or abridged listings of a VTOC by using the LISTVTOC command of the OS/390 IEHLIST utility program. The DFSMSdss PRINT command can also be used to print all or part of the VTOC. A VTOC listing can be obtained under VSE by executing the LVTOC utility program. In both environments, DITTO/ESA′s DVT function may also be used.

5.5.3 Indexed VTOC Considerations (OS/390)

OS/390 has a facility for improved DASD VTOC performance called the Indexed VTOC. Indexed VTOCs are optional but are strongly recommended in OS/390. Their major benefit is improved VTOC performance by avoiding lengthy hardware keyed searches of the VTOC, which tie up the channel and device, and by managing free space information in such a way that the number of I/O operations required to obtain or release space on the volume is reduced. We recommend that you use indexed VTOCs. Details can be found in *DFSMSdfp Advanced Services*, SC26-4921.

The VTOC index itself is a specialized data set which resides on the same volume as the VTOC to which it refers. As such, it has a Format-1 DSCB in the VTOC which contains the index′s data set name and extent information. The index data set name must adhere to the naming convention ″SYS1.VTOCIX.xxxxxxxx″ where xxxxxxxx is user defined: It is recommended that you include the volume′s serial number.

It is not recommended that OS/390 volumes with Indexed VTOCs be used on VSE systems. If the need exists, care must be exercised - each volume has to be converted from an Indexed VTOC to a non-indexed VTOC (that is, a VTOC with no index) before transporting the volume to the VSE system. Otherwise, serious errors may result when the volume is returned to the OS/390 system; that is, VTOC changes made on the VSE system not causing reconstruction of the VTOC are not recorded in the index and, in effect, invalidate the index.

For more information on the Indexed VTOC facility, see *DFSMSdfp Advanced Services*, SC26-4921. For more information on creating Index VTOCs, see the *Device Support Facilities User′s Guide and Reference*.

5.6 VSAM Differences

5.6.1 Introduction

This section covers the differences between OS/390 VSAM and VSE/VSAM. In OS/390, the functions of VSAM and other OS/390 access methods are provided by the OS/390 Data Facility Product (DFP). The term ICF refers to the Integrated Catalog Facility. The term AMS refers to Access Method Services and/or the program IDCAMS. All data set references are to VSAM data sets.

A complete set of DFSMS/MVS publications can be found in the *DFSMS/MVS Library Guide*, GC26-4902 and *DFSMS/MVS General Information*, GC26-4900.

Planning information for Data Facility Systems Managed Storage (DFSMS) can be found in the following manuals:

- *DFSMS/MVS General Information*, GC26-4900
- *DFSMS/MVS Planning for Installation*, SC26-4919
- *Implementing System-Managed Storage*, SC26-3123
- *Get DFSMS FIT: Fast Implementation Techniques*, SG24-2568
- *DFSMS FIT: Fast Implementation Techniques Process Guide*, SG24-4478
- *DFSMS FIT: Fast Implementation Techniques Installation Examples*, SG24-2569
- *RACF General Information*, GC23-3723
- *Getting Started with DFSORT*, SC26-4109

5.6.2 OS/390 Catalogs

At the time of writing, there are three catalog types supported in OS/390. After the end of 1999, only ICF catalogs will be supported in the OS/390 environment. The current catalog types are:

- **ICF catalogs - are the recommended catalog structures for your OS/390 system.**
- VSAM catalogs - these are similar and mostly compatible[5] to VSE/VSAM catalogs. They do not provide the performance and recoverability of ICF catalogs. They are not recommended for normal OS/390 production operations. However, they do provide catalog portability between VSE and OS/390. If used, they should be converted to ICF catalogs when the migration is completed. Details on converting to ICF catalogs can be found in *Managing Catalogs*, SC26-4914, chapter 9.

  The VSAM catalog stores dates with two-digit years. VSE/VSAM has implemented a sliding window interpretation of those dates. OS/390 (DFP) has not made similar changes, and thus dates in VSAM catalogs will not be correctly interpreted in OS/390 systems. Because of this, OS/390 (DFP) has announced that VSAM catalogs will not be supported after December 31st 1999.

[5] Compatibility between VSE/VSAM catalogs and OS/390 VSAM catalogs is discussed in the section entitled 5.6.4, “OS/390 - VSE/VSAM Catalog Compatibility” on page 117.

- OS CVOL catalogs - these are a carry-over from the past (pre-OS/390) and are non-VSAM in structure. It has also been announced that OS CVOL catalogs will not be supported after December 31st 1999. For these reasons, they should ***not*** be implemented in your OS/390 system.

5.6.2.1 Integrated Catalog Facility (ICF)

The architecture of an ICF catalog is quite different from that of a VSAM catalog. An integrated catalog facility catalog consists of two separate kinds of data sets: a basic catalog structure (BCS) and a VSAM volume data set (VVDS). The BCS can be considered the catalog, whereas the VVDS can be considered an extension of the VTOC.

The BCS is a VSAM key-sequenced data set that contains the following information:

- VSAM data sets
  - volume, security, ownership, and association information
- Non-VSAM data sets
  - volume, ownership, and association information

The VVDS is a VSAM entry-sequenced data set. Its name must be SYS1.VVDS.Vvolser. A VVDS resides on every volume that has VSAM or SMS-managed data cataloged in an ICF catalog. It contains the data set characteristics, extent information, and the volume-related information of the VSAM data sets cataloged in the BCS. For SMS-managed non-VSAM data sets, the VVDS also contains data set characteristics and volume-related information.

If you use a function such as DITTO/ESA′s DID command or the ICKDSF REFORMAT command to change a volume serial number (or volser), only the VOL1 label is changed. Since the REFORMAT command or DID command only changes the volser, it does ***not*** change the VVDS name. If a REFORMAT or DID command is used to change the volser, the VVDS name will no longer adhere to the required data set name format of SYS1.VVDS.Vvolser. If the VVDS name does not adhere to the required name format, access to VSAM data sets on that volume has been lost. You won′t be able to RECATALOG these data sets. If your VSE system procedures depend on the use of this capability, those procedures will have to be redesigned.

In the OS/390 environment you should define all your catalogs to be ICF catalogs. ICF catalogs are generally superior in performance, virtual storage savings, recoverability, and space utilization when compared to VSAM catalogs. In addition, many new functions, such as VSAM compression, are only available for SMS managed data sets, which requires use of ICF catalogs. ICF catalogs are not compatible with VSE. You cannot access an ICF catalog from a VSE system.

OS/390 ICF catalogs do not own volumes. Thus, it is possible to have OS/390 VSAM data sets on a given volume that are cataloged in different ICF catalogs. This implies changes to backup and recovery procedures commonly used in VSE/ESA installations.

If a catalog is damaged, restore a backup of the catalog and perform a forward recovery to bring it back into sync. Tools are available for catalog forward recovery such as the Integrated Catalog Facility Recovery Utility (ICFRU). If a cluster is damaged, the old version can be deleted, uncataloged, restored from a backup and then forward recovery can be used to make the cluster data current.

If access to a disk volume is lost, DFSMShsm can be used to perform a full-volume restore with update.

You specify ICF catalog format by including the ICFCATALOG keyword in the AMS DEFINE MASTERCATALOG or DEFINE USERCATALOG command. ICFCATALOG is the default when defining a catalog in OS/390.

VSAM data sets cataloged in ICF catalogs have the UNIQUE attribute -- their space is not owned nor managed by VSAM. Since OS/390 has an integrated Direct Access Device Space Manager (DADSM), space allocation is done through normal JCL and DADSM facilities.

Should your migration plan have a special requirement where a VSE program (prior to its conversion) has to access data that is under control of OS/390, then the data set containing this data should not be cataloged in an ICF catalog. Rather, you should then have a VSAM catalog (described in the next section) for cataloging this data set. Once the VSAM program is converted to OS/390, the VSAM catalog should be converted to an ICF catalog.

5.6.2.2 VSAM Catalogs

VSAM catalogs are supported under OS/390 but are not recommended. These catalogs can be defined to be identical in structure to VSE/VSAM catalogs. (Although identical in structure, the OS/390 VSAM catalogs contain additional information about the data sets that are defined in them; for example, data set and device attributes, data set use statistics.) OS/390 VSAM user catalogs can be accessed by a VSE system if proper measures are taken to assure catalog integrity. See 5.6.4.1, “Accessing a VSE/VSAM Catalog from an OS/390 System” on page 118.

Note that this ability of OS/390 (DFSMSdfp) VSAM to process VSAM catalogs will not be available after December 31st 1999.

Item G023729

Last updated....: 10/13/1997

Abstract.....: WSC FLASH 9741

VSAM CATALOG AND CVOL SUPPORT ENDS IN YR2000

Access to an MVS or OS/390 non-ICF VSAM catalog or CVOL will not be possible after 1999.
The following text was taken from the DFSMS/MVS 1.4 availability announcement made on June 6, 1997 (297-192):

″NOTE: On October 31, 1995, IBM announced Year 2000 Support which stated that, for any S/390 platform running MVS or OS/390 to be considered as Year 2000 Ready, all data sets must use Integrated Catalog Facility (ICF).

It will not be possible to access data via a VSAM Catalog or CVOL when the system date changes past December 31, 1999.
This means that MVS customers who still have data sets cataloged in OS/VS Control Volumes (CVOLs) or in the old VSAM catalogs will need to migrate these to ICF catalogs before the end of 1999.
We look forward to all of our MVS customers taking advantage of the better performance and integrity of ICF. Those VSE VSAM customers who previously shared VSAM data sets between the MVS platform and VSE or VM platforms will need to reevaluate methods for satisfying their data sharing requirements. For further information, refer to Software Announcement 295-464.″

APARs OW25632 and OW25988 and their associated PTFs have been written to assist customers in determining whether or not they are accessing VSAM catalogs. The APARs add two new reason codes to message IEC331I return code 4:

″33 - Explanation: A VSAM catalog has been opened for use by the catalog address space. The usage is accepted.

Programmer Response: VSAM catalogs may not be opened as catalogs beginning 1/1/2000. This message is provided to simplify identifying whether any VSAM catalogs are still in use. They should be converted to ICF catalogs as soon as possible.″

Figure 8 (Part 1 of 2). Extract from WSC Flash 9741

When the system date is on or after 1/1/2000, the following reason code will be issued:

″34 - Explanation: An attempt was made to open a VSAM catalog for use as a catalog. The request was denied.

Programmer Response: VSAM catalogs may not be used beginning Jan 1, 2000.″

Please note that CVOL support will also be removed effective 1/1/2000 but as yet no way to provide warning messages has been identified.

Customers running operating environments prior to DFSMS/MVS 1.4 who have not installed the appropriate maintenance will receive no warning message when processing VSAM catalogs and its entries, but are still subject to errors. Any customer failures resulting from attempts to process VSAM catalogs or CVOLs on OS/390 and MVS systems after December 31, 1999 will not be addressed by IBM service.

Information on how to convert VSAM catalogs and CVOLs to ICF catalogs can be found in chapter 9 of the DFSMS/MVS *Managing Catalogs*, SC26-4914.

Figure 8 (Part 2 of 2). Extract from WSC Flash 9741

5.6.3 OS/390 Catalog Management

5.6.3.1 OS/390 Master Catalog

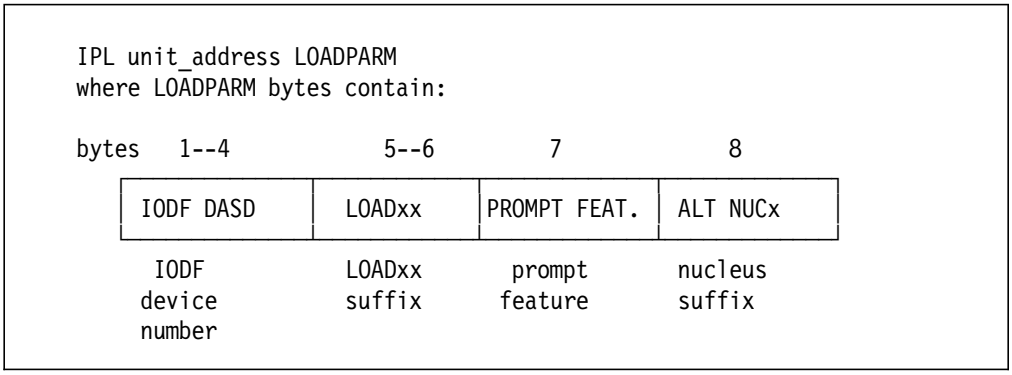
OS/390 requires a master catalog in order to IPL. The master catalog cannot be disconnected and should not normally be ported to another system environment. The OS/390 master catalog should contain only:

- Alias definitions
- catalog entries for system data sets
- pointers to user catalogs

Certain system data sets must be cataloged in the master catalog in order to IPL. System data sets normally have data set names which start with ″SYS1″ as their high-level data set name qualifier. Examples are SYS1.LINKLIB and SYS1.PROCLIB. See the *OS/390 MVS System Data Set Definitions* manual for the names and uses of OS/390 system data sets.

At IPL time the system locates the master catalog via the LOADxx member of the OS/390 system parameter data set, SYS1.PARMLIB. This member contains the master catalog′s data set name, volume serial number, and device type.

If multiple LOADxx members exist (each with a unique ″xx″ suffix), it is possible to choose an alternate master catalog at IPL time. Each LOADxx member would point to a different catalog. This might be done for testing or for backup purposes. The operator can specify the LOADxx member during IPL using the information contained in the following figure:



See *Managing Catalogs*, SC26-4914, chapter 2. You must ensure that all OS/390 data sets required for IPL are cataloged in all catalogs that might be used as an alternate master catalog. Other than content, there is no difference between a user catalog and the master catalog. A catalog is the OS/390 master catalog by virtue of the fact that it was designated as such during IPL.

The master catalog must be on a volume that is mounted and available at all times. The master catalog should be password protected or secured via RACF. This is to insure that user data sets are not cataloged in the master catalog and to insure that end users cannot uncatalog critical system data sets.

Formerly, a single catalog could not serve as the master catalog for more than one system at a time. However, a master catalog could be accessed from another system as a user catalog.

You can now share a master catalog between multiple systems. See *Managing Catalogs*, SC26-4914, chapter 2.

With the introduction of MVS/SP 5.1.0, an installation that has multiple MVS images can share a master catalog and share an IPL volume among multiple MVS images. The system data sets, SYS1.LOGREC and SYS1.STGINDEX are no longer fixed named and unable to be shared. They can now be shared and specified by the installation. In addition, a system symbolic, &SYSNAME, was introduced and can be used as part of data set name specifications for some parameters in PARMLIB. When you use &SYSNAME, data set name specification becomes flexible and you do not need a separate parameter specification for each system in the sysplex.

For example, we can specify the following LOGREC=SYS1.LOGREC.&SYSNAME. The symbolic name, &SYSNAME can also be used in other PARMLIB parameter specifications. You can use &SYSNAME for IEASYSxx parameters VIODSN=, PAGE=, SWAP=, DUPLEX=, and NONVIO=. You can use &SYSNAME for SMFPRMxx parameters DSNAME= and SID=. Catalog if it has been connected to the second system′s master catalog via the AMS ″IMPORT CONNECT″ command.

5.6.3.2 OS/390 User Catalogs

User catalogs contain data set information for user data sets. They should be created as ICF catalogs. VSAM user catalogs may be accessed during the migration period, but should only be used if VSE access is required. They should be converted to ICF catalogs once the migration is over.

The data set names of the user catalogs are contained in the OS/390 master catalog. Information necessary to locate the user catalogs is also defined in the master catalog. The high-level-qualifiers of data sets that are to be cataloged in each user catalog are also identified in the OS/390 master catalog.

OS/390 data set names are divided into qualifiers. The data set name consists of up to 44 characters, grouped into qualifiers of up to eight characters each, separated by periods. For example ″DEPT1.PAYROLL.YTD.DATA″ contains four qualifiers. The high-level qualifier consists of the characters before the first period in the data set name. It is also known as the catalog ″ALIAS″ name, because it determines which user catalog OS/390 will search to find the data set information. Thus the master catalog normally has many ALIAS name ″pointers″ to multiple user catalogs. An OS/390 user catalog may have many ALIAS names associated with it. Each TSO user name will be associated with a particular user catalog through an alias which must be defined for it.

The following diagram illustrates the master catalog to user catalog structure relationship.

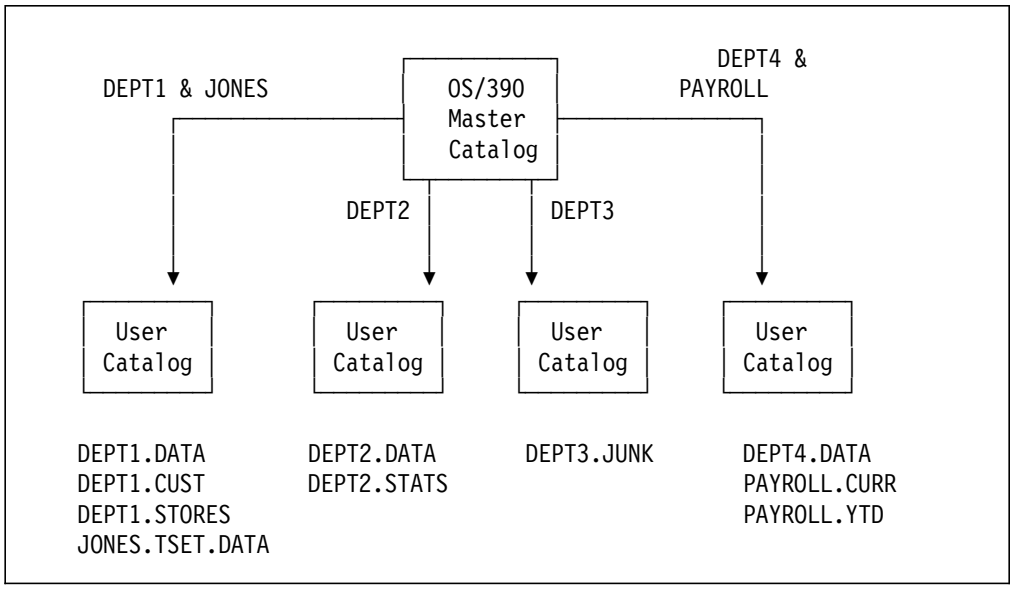


Figure 9. OS/390 Master and User Catalog Structure

The system searches for data sets by finding, via the master catalog, in which user catalog the data set is cataloged. For example, for a data set named DEPT1.PAYROLL.YTD.DATA, the master catalog would direct OS/390 catalog management to the appropriate user catalog (identified by the DEPT1 ALIAS name), and the data set information would then be retrieved from this user catalog. The manual, *Managing Catalogs*, SC26-4914 contains more details about the use of catalogs. It discusses aliases, catalog search order and so on, in Chapter 2.

Do not use JOBCAT or STEPCAT statements in OS/390

In predecessors of today′s OS/390 systems, it was not uncommon to use JCL DD statements specifying user catalogs to be used for a job or a step (JOBCAT or STEPCAT). This procedure was obsoleted with the advent of the Integrated Catalog Facility and its ALIAS mechanism, and it should not be used.

The OS/390 JOBCAT and STEPCAT DD statements when used in jobstreams, bypass the normal catalog search technique via the ALIAS. Their use can cause incorrect cataloging of duplicate ″new″ data sets. In addition, these statements are not supported by Data Facility Systems Managed Storage (DFSMS). JOBCAT and STEPCAT should not be used when migrating to OS/390.

5.6.4 OS/390 - VSE/VSAM Catalog Compatibility

Note

Even though (non-concurrent) sharing of VSAM between VSE and OS/390 is described below, it is recommended that this practice not be extended past the migration period. As VSAM changes are probable with new releases of OS/390 DFP or VSE/VSAM, ″flip flopping″ of data sets and catalogs between VSE and OS/390 should be limited to the migration period.

Note that VSAM catalog dates are stored with a two-digit year representation. Although VSE/VSAM has been updated to interpret these dates using sliding window technologies, DFP/VSAM has not been changed similarly. DFP/VSAM use of VSAM catalogs will not function correctly after December 31st 1999.

See the WSC flash article Figure 8 on page 113.

A VSE/VSAM master or user catalog can be designed (DEFINEd) to be compatible and therefore accessible by an OS/390 system. ***Only VSAM user catalogs may be ported between environments***. For a VSE/VSAM catalog to be compatible (and used in an OS/390 system), the catalog has to have the following characteristics:

1. The catalog was defined with the IMBED option.

   This is the VSE/VSAM default, but many users specify NOIMBED for space savings within the catalog itself.
2. The catalog is on a device supported by OS/390.

   Generally, the disk devices not supported by OS/390 are FBA Direct Access Storage Devices.

Note that non-VSAM data sets within VSAM data spaces, such as VSE/VSAM-managed SAM files, cannot be accessed in OS/390.

5.6.4.1 Accessing a VSE/VSAM Catalog from an OS/390 System

Your migration plan might include the requirement to access VSE/VSAM catalogs from the OS/390 system. ***Under no circumstances should you attempt to share VSAM catalogs or data sets between OS/390 and VSE/VSAM concurrently. There is no system data integrity provided for concurrent access sharing. Loss of data may occur.*** If OS/390 access to a VSE/VSAM catalog is necessary, the following procedures should be used:

1. Execute the VSE AMS command EXPORT DISCONNECT to disconnect the VSE user catalog from the VSE master catalog.
2. Execute the OS/390 AMS command IMPORT CONNECT to connect the VSE user catalog to the OS/390 master catalog.
3. An OS/390 AMS DEFINE ALIAS command may be necessary to reestablish OS/390 ALIAS name structures for that user catalog.

The catalog and data are now ready to use with OS/390 applications. (Note: Suballocated VSAM space may be accessed through OS/390 VSAM catalogs. However, OS/390 VSAM catalogs do not support non-VSAM data sets in VSAM space; that is, a VSE facility usually used for VSE/VSAM-managed SAM files.)

″Reversing″ the above procedure allows VSE access to the catalog once again. This is accomplished by:

1. Execute the OS/390 AMS command EXPORT DISCONNECT.
2. Execute the VSE AMS command IMPORT CONNECT.

   The catalog and data are again ready to use with VSE applications.

DISCONNECT does not prevent access by VSE of OS/390 applications which already have access to the catalog or data sets defined in the catalog. Care must be taken in implementing procedures to make sure the catalog is properly closed in OS/390. In VSE, catalogs are never actually closed, but can be disconnected to prohibit access. In OS/390, you can find out which catalogs are open by using the MODIFY CATLOG command. See the manual *Managing Catalogs*, SC26-4914 for more information.

You may also access an OS/390 created VSAM user catalog from a VSE system by disconnecting the catalog from the OS/390 system and then connecting it to the VSE system. The procedure is the same as that described above.

Under no circumstances should you attempt to share VSAM catalogs or data sets between OS/390 and VSE/VSAM such that more than one system can perform updates concurrently. Read-only sharing may be permitted. See 5.6.6.3, “Cross-System and DASD Sharing” on page 129 for more detail.

5.6.4.2 Converting VSE/VSAM Catalogs to OS/390 ICF Catalogs

The OS/390 AMS CNVTCAT command converts a VSAM catalog to an OS/390 ICF catalog. ***After*** you are sure that VSE will not need to access the VSE catalog(s), they may be converted to ICF format.

Non-VSAM files in VSAM managed space must be deleted before running CNVTCAT. If the VSAM catalog has VSAM clusters defined in sub-allocated space, they will be converted to UNIQUE space during the conversion. All volumes owned by the catalog should be backed up before running CNVTCAT. See *Managing Catalogs*, SC26-4914, chapter 9 for more detailed information.

5.6.4.3 Moving a VSAM Catalog to a Different DASD Type

VSE/VSAM provided no facility for moving a catalog to a different device type or volume serial number. OS/390 VSAM provides this facility for ICF catalogs via the AMS REPRO and EXPORT/IMPORT commands. Using the REPRO function, you may specify the old catalog as the input and the new catalog as output. The new catalog must already have been defined but it may be a different DASD device type and it *must* be a different volume serial number. Moving the catalog does not move the data sets that exist on the old catalog volume.

REPRO has two options, MERGECAT and NOMERGECAT. MERGECAT will transfer entries from one catalog to another, deleting the entries from the source catalog. If there is a failure during MERGECAT, the target catalog contains the only valid entries for some of your data sets. For this reason, do **not** delete the target catalog simply because the MERGECAT failed. If the failure was caused by errors external to catalog management and access method services, simply rerun the REPRO MERGECAT job.

NOMERGECAT will transfer entries to the target catalog but **not** delete them from the source catalog. Therefore, data set entries will exist in both catalogs. If there is a failure during the process, you cannot simply resubmit the job because REPRO NOMERGECAT copies the source catalog into an **empty** target catalog. During the REPRO process, catalog pointers in VSAM VVR entries will be changed to point to the target catalog. Before you restart the REPRO NOMERGECAT, steps must be taken to remove the duplicate data set entries that are now in the source catalog and change the VSAM VVR pointers back to the source catalog. After a REPRO is run, the target catalog is meant to be used as the active catalog.

You could also use EXPORT/IMPORT to move an ICF catalog to another device but you cannot change the catalog name during the process. You can use EXPORT/IMPORT or REPRO to move data sets. Before using REPRO on ICF catalogs, you should refer to *Managing Catalogs*, SC26-4914, chapter 4 and *DFSMS/MVS Access Methods Services for ICF* SC26-4906 for further details.

5.6.5 VSAM Functional Differences

5.6.5.1 Areas of Consideration

The following list summarizes VSE/VSAM functions that are either not supported by OS/390 VSAM or are implemented in a significantly different fashion. Each item is discussed in the text following this list.

- FBA DASD (a general change, not specific to VSAM)
- Catalog structures
  - NOIMBED option
  - Shared volume ownership
- AMS commands
  - CANCEL command
  - DELETE IGNOREERROR
  - SYNCHK parameter
  - XXL KSDS (new in VSE/ESA 2.3, greater than 4GB KSDS)
  - COMPRESS (new in VSE/ESA 2.2, VSAM record compression)
- VSAM CISIZES and blocksizes

- VSE/VSAM-managed SAM files
  - Default models
  - NOALLOCATION data sets
  - Implicit JCL DEFINE
- Reusable data sets
- Partition independent file names
- VSE/VSAM BACKUP/RESTORE and VSE FASTCOPY
- IKQVDU - volume cleanup
- IKQVCHK - catalog check
- Space classes
- VSAM SHAREOPTIONS (SHR(4) and SHR(4 4) differences)

5.6.5.2 FBA DASD

Fixed Block Architecture (FBA) DASD devices such as the 3370, 3310, 9332, 9335, 9336, and FBA virtual disks are not supported by OS/390. Any data sets on these devices must be moved to an OS/390 supported Count Key Data (CKD) or Extended CKD (ECKD) DASD device such as the 3390, 3380, 3350, or 3375. This is generally done by copying the data sets to tape and loading them down with appropriate OS/390 utilities. For VSAM data sets, AMS REPRO is recommended.

5.6.5.3 Catalog Structures

NOIMBED Option

VSE/VSAM catalogs may be defined with the NOIMBED option. This saves DASD space by not replicating the sequence set in the data portions of the catalog. It may make a catalog search take longer an a CKD device. These catalogs cannot be accessed by OS/390 VSAM.

The procedures for converting these back to IMBED format are described in the *VSE/VSAM Programmer's Reference*, SC24-5145. Briefly, EXPORT all files, delete all VSAM space, delete the catalog, redefine the catalog with the IMBED option, re-define the VSAM space, and IMPORT the files. If VSE/VSAM BACKUP is used instead of EXPORT, then RESTORE is used instead of IMPORT. BACKUP and RESTORE should be both easier to use and faster than EXPORT and IMPORT.

The above paragraphs are talking about a VSAM catalog being accessed by both VSE and OS/390 but when the catalog is converted, NOIMBED and NOREPLICATE are recommended when behind cache. IMBED is not recommended for VSAM data sets or ICF catalogs when these data sets are stored on DASD volumes behind a cached control unit because IMBED implies replication and replicated data can result in poor cache usage. If sufficient real and virtual storage is available, and appropriate BUFNI values are specified to keep the index components for VSAM clusters in storage, NOIMBED and NOREPLICATE is also indicated, even with non-cached DASD hardware.

Shared Volume Ownership

VSE or OS/390 VSAM catalogs may own space on more than one DASD volume. Only one VSAM catalog may reside on a volume. Both VSE and OS/390 VSAM

maintain the ″VSAM Ownership Bit″ in the VTOC, and the list of volumes owned by the catalog.

Under VSE, multiple VSAM catalogs can own space on the same DASD volume, as long as only one recoverable catalog owns space on that volume. This support has been provided by adding a VSE unique ″bit map″ in the VSAM catalog, identifying the space that is owned by a particular catalog, on a particular volume. This is not supported with OS/390 VSAM. Only one VSAM catalog can own VSAM space on a volume. ***While OS/390 will prevent OS/390 jobs from creating this environment, it may not recognize shared volumes which have been imported from VSE. This can lead to catalog or data set damage if space is deleted, defined, or expanded under OS/390***. Refer to the *VSE/VSAM Programmer's Reference*, SC24-5145, Chapter 10 for additional information.

OS/390 ICF catalogs do not ″own″ volumes. ICF uses Basic Catalog Structure (BCS) data sets, one per catalog, to contain catalog information. It uses VSAM Volume Data Sets (VVDS), one per DASD volume, to contain information about the VSAM data sets on that volume. Thus it's possible to have multiple VSAM data sets on the same volume that are cataloged in different ICF catalogs. It is also possible to have more than one ICF catalog on a volume.

5.6.5.4 AMS Commands

DELETE IGNOREERROR

VSE/VSAM provides the ″DELETE... IGNOREERROR″ command to DELETE a cluster that had been only partially deleted previously. OS/390 VSAM provides this function for ICF catalogs using the ″DELETE... TRUENAME″ command. Also, the OS/390 DELETE VVR and DELETE NOSCRATCH commands can be used to delete partial VSAM structures.

SYNCHK Parameter

OS/390 AMS does not support the SYNCHK parameter. This was used to test VSE AMS commands by allowing IDCAMS to syntax check the commands without executing them. VSE AMS also supports other test and debugging parameters which may not be applicable in the OS/390 AMS environment.

XXL KSDS (New in VSE/ESA 2.3, greater than 4GB KSDS)

OS/390 VSAM has provided support for VSAM data sets larger than 4GB in size only for SMS managed data sets in extended format with the extended addressability attribute. These data sets must be cataloged in ICF catalogs because they are SMS managed. VSE/ESA Version 2.3 includes new VSAM support for KSDS files only which are larger than 4GB. The implementations of the two VSAM systems are not compatible, due to the differences between VSAM catalogs used by VSE, and ICF catalogs used by OS/390. XXL data sets defined in VSE will have to be unloaded from VSE and reloaded in OS/390.

COMPRESS (New in VSE/ESA 2.2, VSAM Record Compression)

OS/390 VSAM has provided support for compression of VSAM data sets only for SMS managed data sets in extended format and Compaction set to Yes. These data sets must be cataloged in ICF catalogs because they are SMS managed.

VSE/ESA Version 2.2 included new VSAM support for compression of VSAM data sets. The implementations of the two VSAM systems are not compatible, due to the differences between VSAM catalogs used by VSE, and ICF catalogs used by OS/390. COMPRESSED data sets defined in VSE will have to be unloaded from VSE and reloaded in OS/390. REPRO or EXPORT/IMPORT can be used for this unload/reload function.

VSAM CISIZEs and Record Sizes

Both VSE and OS/390 VSAM will select an acceptable CISIZE if none has been specified. If an unacceptable CISIZE has been specified on the DEFINE, both VSAMs will attempt to select an acceptable default. Refer to the manual *DFSMS/MVS Using Data Sets*, SC26-4922, for more information regarding OS/390 VSAM CISIZEs.

You will not be able to directly read a VSE/VSAM KSDS that was created with index CISIZEs that are ***invalid*** for OS/390 VSAM. You must EXPORT or REPRO the cluster from the VSE system and IMPORT or REPRO it into the OS/390 system.

The physical record size is determined during the DEFINE or data set allocation by an algorithm that includes CISIZE and DASD device characteristics. The physical record size will always be equal to or less than the CISIZE. One or more physical records will contain a control interval. Beyond that, VSE and OS/390 use different algorithms.

OS/390 uses one algorithm for data sets that are cataloged in VSAM catalogs, and another for data sets that are cataloged in ICF catalogs. The VSE and ICF algorithms are similar until the CISIZE exceeds 8K. VSE will not create a physical record size greater than 8K for VSE/VSAM releases prior to VSE/ESA 1.3. VSE/VSAM in VSE/ESA 1.3 permits physical record sizes up to 30,720 (30K) bytes, depending on device type and Control Interval size specified. The OS/390 algorithm used with VSAM catalogs will not create a physical record size greater than 4K.

In summary, to be sure you obtain the CISIZE you want, you should explicitly specify the size; that is, not take the default. **For data sets where incompatibilities exist, you will need to EXPORT or REPRO the data set from the originator system and then IMPORT or REPRO it into the destination system**.

VSE/VSAM-managed SAM Files

VSE SAM files in VSAM managed space (SAM/VSAM) are not supported by OS/390. OS/390 cannot access them. They may be converted to VSE or OS/390 Sequential Access Method (SAM or QSAM) data sets. With OS/390, specific track or cylinder addresses for allocation are not required as they are for VSE SAM.

The files may be ported to OS/390 by copying them to a sequential data set (tape or DASD) using a VSE utility such as DITTO. The OS/390 utility, IEBGENER, may be used to copy them under OS/390. Many VSE SAM/VSAM files are used for temporary (work) data sets. These need not be converted as their contents will not be ported to the OS/390 environment.

VSE SAM/VSAM data sets may also be converted to VSAM ESDS data sets. However, this is not recommended as it requires changes to the programs.

Default Models

Both VSE and OS/390 VSAM support the MODEL parameter of the DEFINE command. This allows the attributes of an existing file to be used during the define of a new file. VSE/VSAM also supports three types of model data sets through the AMS DEFINE NOALLOCATION command. OS/390 VSAM does not support the NOALLOCATION parameter. NOALLOCATION is usually used for:

- Through reserved entry names, installation defaults may be specified for one or more VSAM file organizations. This is frequently used for SAM/VSAM files.
- A specific model for a specific data set name. This is typically used to defer allocation until the file is opened. Space is not automatically freed when the file is closed.

NOALLOCATION Data Sets

Through the REUSE option, temporary VSAM data sets (dynamic files) may be defined. No space is allocated until the file is opened and space may be freed when the file is closed. OS/390 supports temporary VSAM data sets in the DFSMS environment.

DFSMS (Data Facility Storage Management Services) is highly recommended and its functions can greatly ease migration of several kinds of VSAM and non-VSAM data sets from VSE to OS/390.

JCL Implicit DEFINE

The default models allow VSE SAM/VSAM files to be defined via JCL, without the need for a specific AMS DEFINE. In OS/390, unless DFSMS is active, all VSAM data sets must be defined using AMS. DFSMS provides new OS/390 JCL keywords which allow some VSAM data sets to be implicitly defined in the JCL.

Reusable Data Sets

Both VSE and OS/390 VSAM support definition of reusable data sets. Usage of reusable data sets differs. VSE allows the REUSE option to be specified in AMS REPRO commands, in the ACB, or through the VSE JCL. OS/390 only supports the options specified in the ACB or through the AMS REPRO command options. OS/390 high level languages (for example, COBOL for OS/390 & VM) permit a reusable data set to be extended if it is opened with the EXTEND attribute, and for sequential access. For older language compilers, a method to extend a reusable data set under OS/390 would be to have the application write to a temporary file, then use an AMS REPRO with the REUSE option to copy it to the intended reusable data set. In neither case can this be controlled by JCL options in OS/390, as it can in VSE.

Two VSE examples are shown below:

```
To extend a reusable data set -
// DLBL .....,VSAM,DISP=OLD
```

- acts as an ACB without RESET (add new records to existing file)
- DISP=OLD overrides IDCAMS REPRO with REUSE

To rewrite a reusable data set -
// DLBL .....,VSAM,DISP=NEW

- acts as an ACB with RESET (delete old records, add new records)
- DISP=NEW overrides IDCAMS REPRO with NOREUSE

Partition Independent File Names

VSE/VSAM partition independent file names start with the character % or characters %%, for example %MY.FILE. These special characters cause the partition identifier (or cpuid and partition identifier) to be appended to the file name when it is DEFINED or accessed. In the example shown, the file name would be MY.FILE.BG if the job was running in the background partition. Access to the file is actually ***dependent*** on the partition in which the job is running. Again referring to the example, the same job running in the Foreground 4 partition would access file MY.FILE.F4.

OS/390 does not have partitions. It has address spaces. An address space does not have an external identifier. Address space independence is automatically provided for all temporary data sets.

VSE partition independent files are frequently used for ″temporary″ work files. They should be converted to OS/390 temporary data sets. An OS/390 temporary data set is specified by a data set name beginning with the character & or the characters &&. OS/390 supports temporary VSAM data sets only with DFSMS. VSE applications that use permanent partition independent files will require another data set naming convention to operate correctly under OS/390.

VSE/VSAM BACKUP/RESTORE and VSE FASTCOPY

IDCAMS BACKUP/RESTORE is used only for VSE VSAM files, while FASTCOPY is used for non-VSAM and full volume backups. FASTCOPY has a stand-alone component. Equivalent functions are provided in OS/390 via Access Methods Services (EXPORT/IMPORT) for VSAM, DFSMS/MVS utilities (IEBGENER, IEBCOPY and so on) for non-VSAM, and the Data Set Services component of DFSMS (DFSMSdss) for data sets (VSAM and non-VSAM) and full volume dumps. Archive tapes created by VSE/VSAM BACKUP/RESTORE or VSE FASTCOPY cannot be processed by DFSMSdss. Tapes produced by VSE/VSAM EXPORT or REPRO may be processed by OS/390 VSAM. Any non-portable archive tapes should be restored to DASD using the appropriate VSE utilities. Then the DASD files should be dumped to tape using a backup technique compatible with OS/390 backup/restore programs such as DFSMSdss. This needs to be done during the OS/390 migration while both operating systems (VSE and OS/390) are available.

IKQVDU - Volume Cleanup

This VSE/VSAM utility does not exist in OS/390 VSAM. The IKQVDU functions ″SCRATCH DSN″ and ″RESET OWNERSHIP″ were used to remove unwanted VSE/VSAM data spaces from a volume and to turn off the VSAM ″Ownership Bit″

in the VTOC. This equivalent function is performed in OS/390 VSAM by the AMS command ALTER REMOVEVOLUMES.

The volume cleanup function of ″ALTER REMOVEVOLUMES″ should only be used when the catalog is not accessible or totally unavailable. This command may also be used to remove candidate volumes as in VSE/VSAM.

IKQVCHK - Catalog Check

This VSE/VSAM utility does not exist in OS/390 VSAM. The AMS DIAGNOSE and EXAMINE commands provide an equivalent function.

Space Classes

Space classes of VSAM data spaces are not supported by OS/390 VSAM. However, VSAM files, data spaces, or volumes established under VSE/VSAM with space classes can be processed by OS/390 VSAM as long as the VSAM catalog is supported (December 31st 1999).

VSAM SHAREOPTIONS

Since VSE and OS/390 use ***totally different implementations*** for data set sharing and control, they provide no protection from each other through any of the VSAM shareoptions. There are significant VSE and OS/390 differences in the access and protection provided by shareoptions three and four.

Shareoptions one and two (SHR(1) or SHR(2)) function exactly the same in VSE/VSAM and OS/390 VSAM. SHR(3) and SHR(4) provide cross-partition (or cross-address space) and cross-system access to VSAM files. These will be discussed in the next section of this chapter.

VSE/VSAM SHR(4) was used by CICS/VSE to allow CICS applications to update a VSAM data set through both the base cluster path and alternate index (AIX) path, prior to the availability of data set name sharing in VSE. Data set name sharing became available in VSE/ESA 1.3. This is not necessary with CICS/OS/390. CICS/OS 1.7 or later uses OS/390 VSAM data set name sharing to allow these updates, with integrity. Note that both the base cluster and AIX(s) must be in the same VSAM LSR buffer pool or use NSR buffer pools.

5.6.6 Data Sharing and Integrity

You should read this section very carefully. There are significant differences in the cross-partition or cross-address space and cross-system protection provided by VSE and OS/390 VSAM shareoptions. OS/390 VSAM SHR(4 x) or SHR(4 4) provides *less* ***automatic protection than VSE/VSAM. Applications that use VSE/VSAM SHR(4) or SHR(4 4) protection or VSE DASD sharing protection must be carefully evaluated in light of the differences in protection provided by OS/390 VSAM.*** For more details, refer to *DFSMS/MVS Using Data Sets*, SC26-4922. In a parallel sysplex environment, Record Level Sharing (RLS) can be used to access VSAM data sets instead of shareoptions. RLS provides complete integrity.

5.6.6.1 Cross-Region Sharing - Single CPU Environment

Whenever a VSAM data set (ACB) is opened by more than one control block structure concurrently, data integrity must be considered. OS/390 VSAM offers two levels of protection and/or sharing within a single CPU.

1. OS/390 will prevent concurrent update/update or update/read access to a VSAM data set if DISP=OLD is coded on the JCL DD statement. If DISP=OLD is specified, the shareoptions will be treated as (1 3). This can potentially provide performance improvements for load jobs.
2. VSAM will monitor access via the data set shareoptions if DISP=SHR has been specified on the DD statement.

The purpose of the VSAM shareoptions is to permit the user to specify the required level of integrity of the data set and prevent possible loss of records, updates, or even total loss of access to the data set. There are actually two types of integrity that can be of concern and the shareoptions vary in the type of integrity provided:

- Write integrity - Assurance that, if an update or add is done, it will not be lost and the data set will not be destroyed.
- Read integrity - Assurance that the record read is current (that is, no other user has since updated it).

Now let us review the shareoptions with respect to the integrity provided.

	Integrity	
	Read	Write
SHR(1 x)	YES	YES
SHR(2 x)	NO	YES
SHR(3 x)	NO	NO
SHR(4 x)	NO*	NO**

Figure 10. OS/390 VSAM Integrity Provided by Cross-Region Shareoptions

* VSE/VSAM will refresh buffers if a data set is open for output. VSE/VSAM also automatically enqueues on CIs and CAs to ensure integrity in the event of concurrent requests. OS/390 buffer refresh is done only for random (direct) reads. OS/390 VSAM does not automatically enqueue on records as VSE/VSAM does.

** VSE/VSAM SHR(4 x) guarantees the write integrity of a VSAM data set. OS/390 VSAM SHR(4 x) does not guarantee the write integrity of a VSAM data set.

With SHR(1 x) a user can not open a file for input to read if another user has opened it for output, so both read and write integrity are assured.

With SHR(2 x) read integrity is not assured because a program (user) may be accessing a data record from a buffer while another user is updating it on disk. Write integrity is assured since there can only be one update at a time.

With SHR(3 x) VSAM does not prevent any user's open and does not monitor their access. Yes, you can destroy the file!

OS/390 VSAM Cross-Region SHR(4)

VSE VSAM SHR(4 x) will refresh buffers from disk for every read I/O, and will also lock the record, CI, or CA as appropriate to protect the file and user data from corruption by possible concurrent update activity. SHR(4) data sets can have inserts or updates which may cause CI and CA splits, and secondary allocations can also safely be handled by VSE/VSAM.

OS/390 VSAM SHR(4 x) works very differently than VSE/VSAM. When SHR(4 x) is specified OS/390 VSAM takes only the following special actions:

OS/390 VSAM will refresh the buffers from DASD for random reads. OS/390 VSAM will write the CI to DASD following the CHECK for random writes. This allows some read and write integrity protection. However, OS/390 VSAM does not automatically enqueue on the record, CI, or CA as VSE/VSAM does. Hence another user may simultaneously update the same record or control interval. Also OS/390 VSAM does not refresh buffers during sequential processing.

When SHR(4 4) is specified, a change in the High-Used-RBA of any component (data or index) is not allowed. For a KSDS, this means:

1. no CA splits are allowed
2. the high-key CI cannot be extended

For all types of data sets, no extensions or new extent allocations are allowed. If a program causes any of the above conditions, it will receive a ″no space″ error code.

These restrictions provide some protection since it is not possible for multiple users to cause concurrent CA splits or extend the data set. However, without user programming of Assembler ENQ/DEQ macros it is still possible to:

1. lose updates
2. read back-level records
3. cause data set failures during concurrent CI splits

Of course, even with user ENQ/DEQ programming, it is possible for errors to cause the same loss of data conditions.

Control Block Update Facility (CBUF) is used if SHR(3 3) or SHR(4 3) is used with JCL DISP=SHR. This removes the programming restrictions related to updating the High-Used-RBA, stated above. It does not assure full read or write integrity. With SHR(4 3), buffers are refreshed for each direct request.

Note

To provide equivalent VSE/VSAM SHR(4 x) protection, when multiple users are updating the same data set from different address spaces, the OS/390 VSAM SHR(4) user should read the chapter entitled ″Sharing a VSAM Data Set″ in the manual *DFSMS/MVS Using Data Sets*, SC26-4922.

In addition, partial to complete solutions for this functional difference between VSE and OS/390 are available from software vendors which provide functions similar to VSE/VSAM SHAREOPTION(4).

5.6.6.2 Single Region Data Set Sharing

Single ACB Open - Multiple String Processing

Full write integrity is provided within a single region provided the user uses a single ACB to process the data set. In high level languages an ACB equates to:

1. a SELECT statement in COBOL
2. a file DECLARE in PL/I
3. an ″F″ statement in RPG II

If multiple ACBs (or high level language equivalents) are used, the protection of shareoptions must be relied upon unless DSNAME sharing is used (both COBOL and PL/I always use it). See “Intra-Region Data Set Name Sharing” below. Multiple positions may be maintained in the file via use of multiple strings (that is, the ACB STRNO parameter). The strings may be used for multiple requests from the main task or its subtask. VSAM will automatically provide exclusive control protection for output requests and read integrity if ″GET for UPDATE″ is used.

Within the same region the data set can be updated concurrently (even with DISP=OLD) and VSAM ensures integrity because a single ACB control block structure is used.

On a GET UPDATE or PUT request, VSAM acquires exclusive control of the CI, after checking that no other string is accessing the CI. Any string which wants to make an update or add to the same CI, will get an ERROR CODE = X′ 14′ with R15 = X′08′. The exclusive control ends when the subtask possessing it issues a GET UPDATE for a record in another CI or issues an ENDREQ or issues a PUT UPDATE for a record read previously by a GET UPDATE. The exclusive control does not impede simple READs for the other subtasks. The user requiring read integrity must specify UPD intent on all RPLs.

Both CICS/OS and IMS/VS DC use multiple string processing with a single ACB structure. They intercept the VSAM exclusive control error codes and suspend or wait the task until the requested resource is available.

Intra-Region Data Set Name Sharing

If DSNAME sharing is specified in the ACB (that is, MACRF=(DSN...)), a data set may be accessed from multiple ACBs within the same region. VSAM assures integrity because there is only one control block structure. This protection is provided because DSNAME sharing tells VSAM to tie the control block structure of the second ACB to the first *if* the data set name matches.

If DSNAME sharing is ***not*** specified in the ACB, the default (DDNAME sharing) applies and VSAM operates as if the data set is being shared by users in different address spaces. The OS/VS COBOL and OS PL/I compilers always use DSNAME sharing when multiple file statements are used.

For VSE users of CICS (since VSE/ESA 1.3), DSNAME sharing has been available as well, so VSE users will have at least the same support in this specific area, and for older VSE installations, OS/390 will provide a significant enhancement.

5.6.6.3 Cross-System and DASD Sharing

You are in a cross-system sharing environment whenever you allow more than one copy of any operating system to access the same DASD volume concurrently. This includes multiple OS/390 guests running under VM or PR/SM. **You must not attempt to update via DASD sharing between VSE and OS/390 systems**. The methods of DASD sharing protection are totally incompatible and you risk contamination or loss of:

- data
- data sets
- catalogs
- VTOCs

Should you be planning to share ICF catalogs or VSAM data sets between ***two or more OS/390 systems***, you should read the chapter entitled ″Sharing a VSAM Data Set″ in the *Using Data Sets* manual. How catalogs are shared is documented in *Managing Catalogs*. The volume the catalogs are allocated on must be defined as SHARED to all images and the shareoptions for the catalog must be 3,4.

OS/390 Global Resource Serialization (GRS) and the new Record Level Sharing feature (part of SYSPLEX support) can provide additional cross-system and DASD sharing capabilities. For more information on GRS, refer to *OS/390 V1R1.0 MVS Planning - Global Resource Serialization*, GC28-1818, or *OS/390 V1R3.0 MVS Planning - Global Resource Serialization*, GC28-1759.

OS/390 Definitions for DASD Sharing Support

In order to provide any protection in a DASD sharing environment you must let OS/390 know that the device may be shared. This is done by specifying the parameter ″SHARED″ in the Hardware Configuration Definition data set. Without this parameter, OS/390 will provide no DASD sharing protection.

OS/390 VSAM Cross-System Shareoptions

Unlike VSE/VSAM, the cross-region shareoption has no meaning for cross-system sharing with OS/390 VSAM ***unless*** OS/390 GRS is used to control cross-region sharing of data sets. What is important is the ***second*** field of the SHR parameter. This field can only be 3 or 4. In other words, only SHR(x 3) or SHR(x 4) is valid for an AMS DEFINE or ALTER.

Record Level Sharing (RLS), available to users of OS/390 systems with Parallel Sysplex capabilities (Coupling Facility, integrated or stand-alone), together with DFSMS support included in DFSMS 1.3 or later provides a much higher level of sharing with protection for OS/390 users. It is an alternative to using shareoptions for accessing VSAM data sets.

RLS can be used in batch environments but there are restrictions. RLS is designed primarily for VSAM data sets used by CICS applications. With VSAM RLS, multiple CICS systems can directly access a shared VSAM data set, eliminating the need for function shipping between AORs and file owning regions (FORs).

Information on Record Level Sharing can be found in several manuals:

- *Planning for Installation*
- *DFSMSdfp Storage Administration Reference*
- *Using Data Sets*

SHAREOPTIONS (X 4)

Cross-system SHR(x 4) provides the same limited protection across systems as cross-region SHR(4). Extensions of the data set's high-used RBA are prohibited. To provide complete integrity protection it is the user's responsibility to write Assembler routines using the RESERVE/RELEASE macros in addition to the ENQ/DEQ macros required for cross-region. This is a complex undertaking. If it was simple and/or if performance was acceptable, OS/390 VSAM would have implemented it. Alternatives to complete DASD sharing of VSAM data sets should be considered first. See *DFSMS/MVS Using Data Sets*, SC26-4922, for greater detail on coding RESERVE/RELEASE routines and the alternatives.

SHAREOPTIONS (X 3)

If SHR(x 3) is used, all data set opens are allowed across systems. OS/390 VSAM provides no protection for the data set.

5.6.6.4 DASD Sharing Considerations

A second system opening a data set that is open for output on another system will receive the OPEN return code 116 (X'74') which indicates the data set was not properly closed. VSAM will automatically issue the VERIFY macro for the program.

Once the data set is open, a cross-system program must contend with the possibility that the data, indexes, extents, the RBA of records, and the High-Used-RBA of the data set may be changing due to updates in another system. This may cause VSAM error codes and/or abends of the programs using the data set.

Alternatives to VSAM Data Set Sharing

Probably the simplest way to avoid the problems of cross-system and cross-region sharing is to schedule all data set access through a single system and address space. This generally means that batch updates must be performed while the files are unavailable to CICS systems.

CICS VSAM data sets can be accessed from multiple CICS address spaces via the CICS Multiple Region Option (MRO) or across systems with Inter-Systems Communication (ISC). However, there is no CICS support for cross-region or cross-system sharing of VSAM data sets with batch jobs.

IMS/VS DC and DB offer methods of cross-system and cross-region sharing of DL/I (IMS) data bases via the IMS Resource Lock Manager (IRLM) and Data Base Resource Control (DBRC). See the appropriate IMS/VS manuals for details.

CICS/OS provides for cross address space sharing of IMS/VS data bases via the CICS Shared Data Base facility.

The DB2 Transparency Feature allows some VSAM files to be loaded into DB2 table spaces and accessed with existing batch or CICS/OS VSAM applications.

This provides cross address space sharing as well as journaling and recovery for the batch applications. It also allows existing files to be accessed with new application tools such as QMF without having to rewrite existing applications.

5.6.7 Programming Languages and VSAM Support

For additional information on program languages and VSAM considerations, see the various language chapters in this publication. In addition, each of the languages has its own publication library generally including migration guides as well as reference manuals. These manuals should be consulted for additional information if necessary.

5.6.7.1 COBOL for OS/390 & VM

IBM COBOL for OS/390 & VM is generally source-compatible (in terms of the VSAM function provided) with IBM COBOL for VSE. Both compilers provide similar support for VSAM KSDS, ESDS, RRDS, Alternate Index (AIX) path processing, and reusable data sets. Chapter 12, “COBOL” on page 249 contains more information.

5.6.7.2 OS/VS COBOL

See Chapter 12, “COBOL” on page 249, for details of DOS/VS COBOL and OS/VS COBOL migration requirements.

5.6.7.3 RPG II

IBM OS/VS RPG II (5740-RG1) is generally compatible with VSE RPG II (5746-RGI) Release 3. However, OS/VS RPG II is not supported with CICS/OS or IMS/VS HLPI. OS/390 VSAM does not allow an empty cluster to be opened for input; VSE/VSAM permits this operation for SAM-ESDS workfiles only.

5.6.7.4 PL/I

PL/I for OS/390 and PL/I for VSE are essentially compatible in terms of VSAM function, and are generally considered source language compatible.

5.6.7.5 Assembler

For a discussion of Assembler programming considerations, see Chapter 13, “Assembler” on page 267.

5.6.8 VSAM Error and Reason Code Compatibility

OS/390 VSAM Error codes and reason codes may have a slightly different meaning than VSE/VSAM. In all cases, OS/390 documentation should be consulted such as the *DFSMS/MVS DFSMSdfp Diagnosis Reference*, LY27-9606. Especially in Assembler, the logic for specific error codes should be verified. In some cases, OS/390 VSAM provides additional information.

5.6.9 DFSORT and VSAM Considerations

Sorting of VSAM files with DFSORT for VSE or VSE SORT/MERGE Version 2 and with OS/390 DFSORT should be compatible with the following two exceptions:

1. Sorting or merging FROM and TO the same (reusable) VSAM data set. (This is often referred to as a suicide sort.)

   This function is supported in VSE DFSORT and VSE Sort/Merge, but not supported in OS/390 DFSORT. A suggested circumvention is to sort the VSAM data set to a temporary data set (either VSAM or non-VSAM) and then

perform a DFSORT COPY function to copy the temporary data set back into the original SORTIN VSAM data set.

2. Sorting multiple VSAM data sets in the same step.

   VSE SORTs allow multiple inputs through multiple SORTIN(n) DLBL or TLBL statements. DFSORT allows only one input to sort, the SORTIN DD statement. Through OS/390 JCL, multiple sequential data sets may be concatenated, but VSAM data sets may not be concatenated. A circumvention would be to use VSAM REPRO to copy the VSAM data sets to one or more temporary sequential data sets. Another circumvention would be to perform multiple sorts each using a single VSAM data set as input to each sort step and then perform a single merge to merge the data sets to produce a single sorted output file. Still another solution would be to write a sort exit that performs the input processing and passes the records to DFSORT.

Chapter 6. CICS

6.1 Introduction

This section is directed to individuals with a working knowledge of both CICS for VSE/ESA and CICS Transaction Server. Without this knowledge, a reader may find this section less fulfilling. Also, you should understand that the scope of this section is to provide general migration tasks and considerations, and should not be considered a replacement for the manuals referenced in this section and/or an individual CICS migration experience.

In the context of this chapter, CICS for VSE/ESA 2.3 is the source subsystem, on which the subject migration activities and consideration are based, although some notes discuss parameters and/or illustrations of pre-CICS for VSE/ESA 2.3 subsystems. Also, references to CICS for VSE/ESA 2.3 may be used interchangeably with CICS/VSE or CICS/DOS/VS.

CICS Transaction Server for OS/390 1.2 is the resultant migrated subsystem, in which all activities/tasks should reside. Please note that references to CICS Transaction Server 1.2 may be used interchangeably with CICS/ESA, and/or CICS.

In the final section CICS with DL/I DOS/VS is discussed. References to DL/I DOS/VS may be used interchangeably with DL/I.

6.1.1 Overview CICS Transaction Server

As an overview, the base CICS element of CICS Transaction Server is CICS 5.2. This element, the CICS successor to CICS/ESA 4.1, is exclusive and includes features and products available with prior CICS versions:

- CICS Web Interface
- Open Network Computing Remote Procedure Call (ONC RPC)
- CICS Transaction Affinities Utility
- CICS-DB2 attachment facility

The non-exclusive elements of the product, also available as separate products, are:

- REXX Development System for CICS/ESA
- REXX Runtime Facility for CICS/ESA
- CICS Distributed Data Management (DDM)
- CICS Application Migration Aid Version 1.1
- CICSPlex SM Version 1.3
- CICS Clients Version 2.0.2
- Transaction Server for OS/2 Warp Version 4 (90-day evaluation copy)

New functions included in this release of the single package solution are:

- Support for single MVS (R) image systems using the DASD logging function of OS/390, provided in OS/390 Version 2 Release 4. Supports single image in sysplex configurations without a coupling facility (non-parallel sysplex) and stand-alone OS/390 systems (single-system sysplex).
- New interface that allows 3270-based CICS transactions to run unchanged without a 3270 terminal.

- Enhanced interface to the World Wide Web (WWW) adds support for 3270-based transactions.
- The CICS Gateway for Java has been ported for execution on OS/390 as an OpenEdition (R) application with CICS TS as the CICS server in a two-tier configuration.
- REXX for CICS (Development and Runtime) added as two new elements of CICS TS.
- Support for S/390 (R) Parallel Sysplex extended with a new system management facility for defining and installing CICS resources across multiple CICS occurrences that are managed by the CICSPlex SM function on S/390 systems.
- New DB2 resource definitions with resource definition online (RDO) as alternative to resource control table (RCT) definitions allowing 7 day, 24 hour availability.
- Added client/server capability, with support for client partner LU6.2 applications across a TCP/IP network.

Key Prerequisites

- OS/390 or MVS/ESA SP Version 5.2 or later
- Either OS/390 Version 2 Release 4 DASD-only logging for single-system sysplex or a coupling facility for Parallel Sysplex

6.1.2 Essential Supplemental Reading and Migration Support Material

One of the critical components to a successful migration is access to all required manuals. Therefore, you are advised to order all CICS Transaction Server for OS/390 1.2 manuals as soon as possible.

For the latest information on what manuals are available with CICS, you should review the *Planning for Installation*, GC33-1789 and *Release Guide for CICS Transaction Server*, GC33-1570.

Pre-CICS for VSE/ESA subsystems migrating to CICS Transaction Server for OS/390 must read prior CICS/VSE Release Guides for possible migration task(s) that may not be addressed otherwise. CICS/VSE 2.3 customers should review the CICS/ESA Migration Guide 3.1, CICS/ESA Migration Guide 3.2, CICS/ESA 3.3 Release Guide, CICS/ESA Migration Guide 4.1, and *CICS Transaction Server Migration Guide*, GC33-1571. Also, your IBM service provider can access the CA1B SupportPac package on the Hursley TXPPACS disk. This package is a CICS/MVS 2.1.2 to CICS/ESA 4.1 migration cookbook, which should give you a perspective of changes to CICS/ESA for MVS users, plus CICS/VSE differences.

For example: An installation migrating from CICS/DOS/VS 1.6 should read the CICS/VSE Release Guides for 1.7, 2.1, 2.2, CICS/ESA Migration Guides 3.1, 3.2, CICS/ESA 3.3 Release Guide, CICS/ESA Migration Guide 4.1, and *CICS Transaction Server Migration Guide*, GC33-1571.

Note: IBM facilitates access to IBM manuals via the INTERNET. Using the INTERNET location:′http://www.ibm.com/′, you can access IBM′s BookServer, your electronic library of books on the World Wide Web.

BookServer allows you to easily manage and display electronic books grouped into catalog collections and bookshelves. BookManager′s high-performance, morphological searching capabilities let you search books and entire bookshelves for the information you need.

Please contact your local IBM Representative for more information on how to access IBM manuals via the INTERNET.

Another useful INTERNET location 'http://www.hursley.ibm.com/cics', is the CICS home page.

The CICS Internet Home page provides a service/support segment that allows easy access to a wide range of material that complements the CICS Family of products. As part of the Service and Support segment, user forums are available for CICS migration tidbits and Qs and As.

Also, the CD-ROM collect kits, can be a useful source for IBM manuals. The *Collection Kit for Transaction Processing and Data Products*, SK2T-0730 includes the unlicensed manuals except for:

- *CICS Master Index*, SC33-1704
- *CICS Transaction Server for OS/390 Licensed Program Specifications*, GC33-1707.

6.1.3 General Compatibility Comments

One of the strengths of the CICS products is the portability of the Command Level API between operating systems. However, applications that include Report Controller API ***cannot*** be migrated and there is no printer and spool file manipulation facilities available in the CICS TS environment. Thus, you should prepare alternate solutions (such as, MVS LAN/RES, user written programs, TCP/IP print daemons, vendor packages) if your users require the same or similar functions or identify the service as no longer available (the sooner the better) to the users.

CICS system facilities will differ between the two operating systems. Facilities based upon the operating system architecture and product-unique (VSE, MVS) functions will have to be re-worked during the conversion to OS/390. One significant change will be the effects on performance and tuning. OS/390 will accommodate additional resources and larger buffer pools resulting in I/O reductions, improved response times, and higher transaction rates. It will also introduce new tuning controls such as the MVS Systems Resource Manager (SRM) that can provide more consistent response times. For these reasons, a stress test of the CICS/MVS system should be a part of the overall MVS migration plan. The major differences between VSE and MVS CICS application programs can be attributed to unique facilities provided by the programming languages or the operating systems. As long as CICS application programs have adhered to standard CICS programming interfaces outlined in the *CICS Application Programmer's Reference Manual*(s) (APRM), the migration or conversion effort should be minimal. In most cases only a CICS/MVS translation and compilation will be required. If the applications are using non-CICS functions such as GET/PUT or COMREG, they may require significant recoding.

6.1.4 Virtual Storage Considerations for MVS

To minimize change during the migration to MVS, the general recommendation is to bring multiple CICS/VSE systems across to MVS the same as they are in the VSE environment. That is, if you have two or more independent production CICS systems under VSE, you would want multiple independent production CICS systems under MVS. MVS/ESA provides considerable virtual storage constraint relief for CICS systems. This is due to the ability to place almost all management modules, and many control blocks above the 16 megabyte line. In addition,

application programs may be placed above the 16 megabyte line if they are written in VS COBOL II, PL/I, C++ or HLASM.

If you have a need to combine systems or you are adding new CICS applications that could introduce additional virtual storage constraints, you should consider the use of CICS Multiple Region Operation (MRO). In addition to providing virtual storage constraint relief, MRO may provide better utilization of multiple processors and improved availability and integrity for your CICS/MVS applications. These benefits are accomplished by the separation of functions into separate address spaces. Note that MRO path lengths will be longer than for a single system image, and there are considerations for operations, recovery, and application design.

A typical MRO system could have one address space for terminal activity (TOR), one for data base activity (FOR), and one or more for application code (AOR). By separating these functions, some problems may be isolated to a single address space (application function). This means that other functions may continue to operate and the time required to restore a lost function may be reduced.

An example of an MRO environment is depicted in Figure 11:

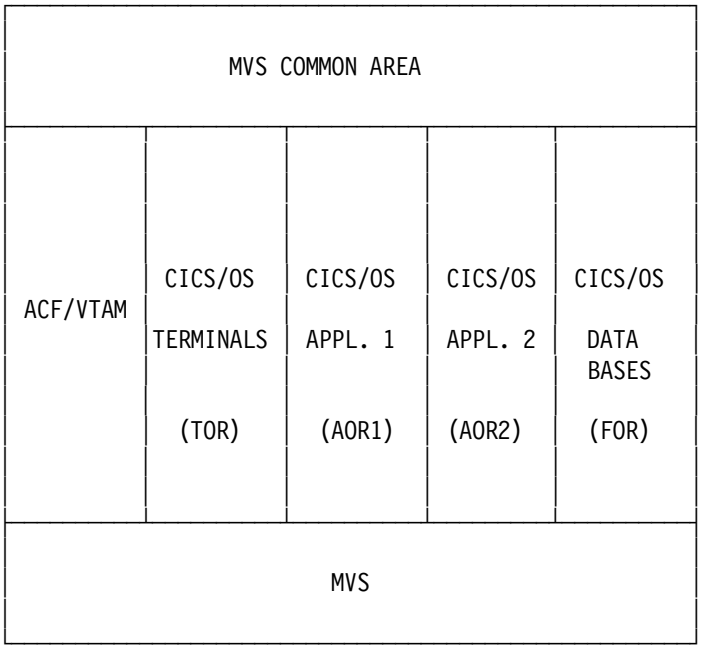


Figure 11. Example of an MVS CICS/OS System using MRO

Note: CICS TS 1.2 also supports VSAM Record Level Sharing (RLS), which may reduce the need for some FORs.

6.1.5 CICS General System Considerations

As part of the CICS/VSE to CICS TS migration there are release enhancements you should know about before the migration; below are a few to consider.

CICS TS products no longer support: Macro-level programs, BTAM devices and controllers, CICS internal security and signon table, Nucleus load table, Application load table, system generation, journaling to tape, shutdown statistics

to SYSLST, access to CICS system control blocks. You should consider what impact each of the removed service or support will have on your migration.

Macro-level programs

You need to convert macro-level programs to command level. Please account for and anticipate the additional CPU requirements for the converted programs.

BTAM devices and controllers

As an alternative consider the supported devices of ACF/VTAM and/or TCP/IP. TCP/IP communications are also supported using either the ONC RPC function or the CICS Web interface. Please review the *CICS ONC RPC Guide*, SC33-1778 and *CICS Internet and External Interface*, SC33-1944 for more detail on the concepts and use of ONC RPC.

CICS internal security and signon table

Since the internal security signon table is no longer supported, you can migrate user data from an existing signon table (SNT) to the RACF database.

If you are using CICS for VSE/ESA 2.3, you can use the security migration aid to assist you with the migration of your CICS internal security definitions to an environment where their resource(s) can be defined with RACF.

Nucleus load table

CICS management modules are restructured into DOMAINS. In the process, CICS removed this function.

Application load table

CICS management modules are restructured into DOMAINS. Thus, CICS removed the possibility of you aligning application programs. You can still specify program residency via the RDO. However, you should review the Resource Definition Guide to get a better understanding as to where this parameter is applicable.

Journaling to tape service

CICS TS Support for single MVS image systems is through DASD logging. DASD logging is for single image in sysplex configurations without a coupling facility (non-parallel sysplex) and stand-alone OS/390 systems (single-system sysplex). Also, CICS TS supports coupling facility logging. The point to remember is that you must review your journaling requirements and operation procedure for CICS TS journal support.

Shutdown statistics to SYSLST

There are numerous changes to CICS statistics records, generally as a result of the new domains created in CICS/ESA, such as the transaction manager domain. As a result, a number of statistics DSECTs, previously supplied as copybooks, are obsolete and withdrawn. Therefore, you should consider alternatives to printing CICS shutdown statistics.

There is a PLT program in SDFSAMP called DFH$STED which can be placed in the startup PLT to stagger writing shutdown or interval statistic to SMF. You should note that without this staggering, you could experience a significant performance problem if interval stats or shutdown occurs in many regions at the same time.

Access to CICS system control blocks

CICS management modules are provided as pregenerated systems for MVS. All the functional areas in CICS are provided completely in object-code only form (OCO), without licensed source materials. Thus, you can not modify the CICS source as in your present releases. Hence, if your present system includes source modification to CICS/VSE management modules, you should evaluate whether the target subsystem addresses your present requirements and/or can CICS exit program interfaces be used to supplement your present modifications.

Figure 12 on page 139 is a layout of CICS restructured management modules called CICS domains. Basically each domain is a single major component of CICS.

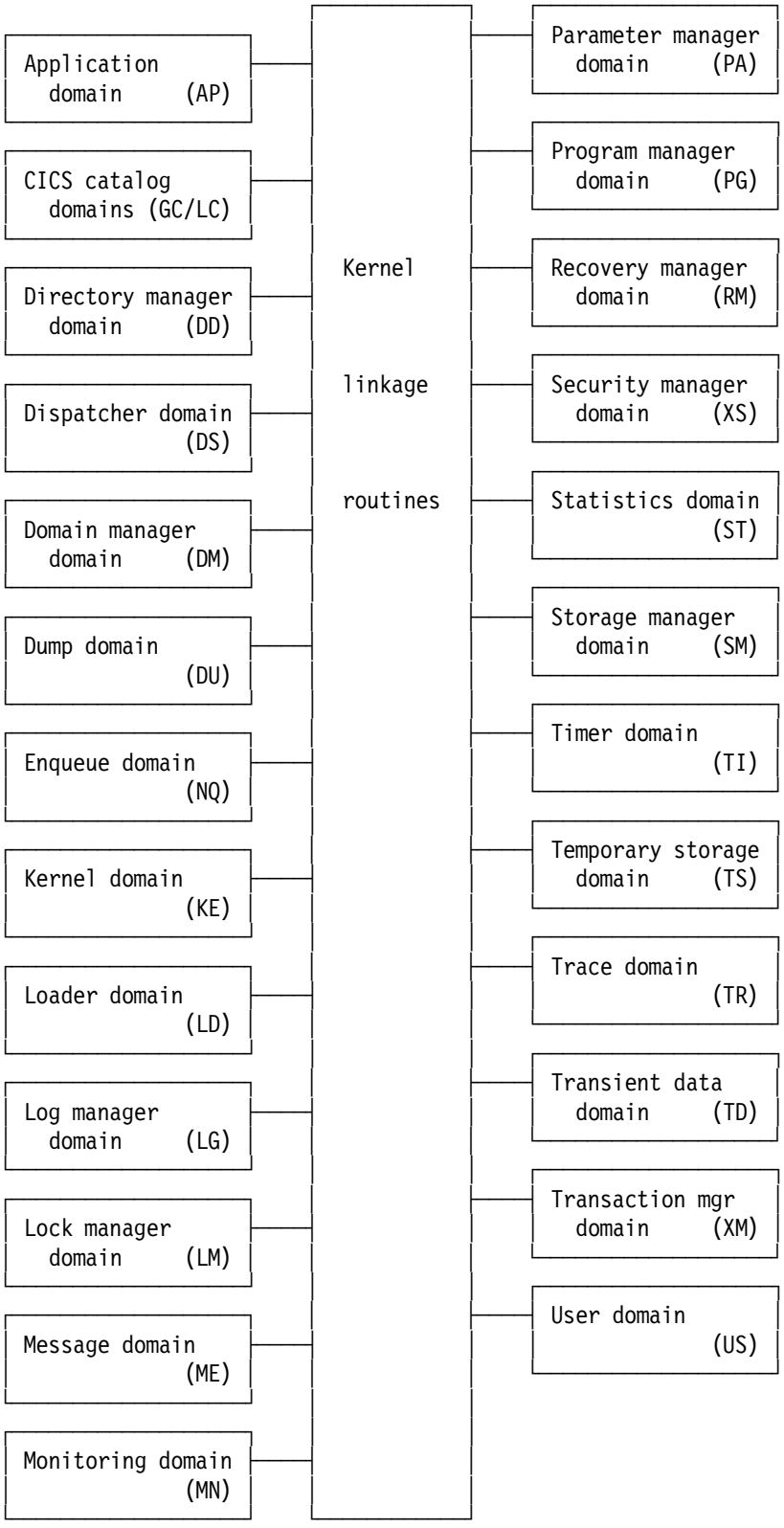


Figure 12. CICS Domains

Note: The application domain is mainly non-OCO, but it contains these OCO components:

CICS data table services
RDO for VSAM files and LSR pools
Some EXEC CICS system programming functions
Autoinstall terminal model manager
Partner resource manager
SAA Communications and Resource Recovery
Some of the file control functions
Recovery manager connectors interfaces.

Domains never communicate directly with each other. Calls between domains are routed through kernel linkage routines. Calls can be made only to official interfaces to the domains, and they must use the correct protocols.

Each domain manages its own data. No domain accesses another domain′s data directly. If a domain needs data belonging to another domain, it must call that domain, and that domain then passes the data back in the caller′s parameter area.

Now with the CICS restructuring in mind, here are a few other general system items to consider with your migration:

- CICS TS recovery manager facility uses the system log for recovery, thus an intermediate data set, such as the restart data set (DFHRSD) is not required for the CICS TS.
- CICS/ESA removed the assembly of a Process Program Table (PPT) and Program Control Table (PCT) from the list of tasks to perform during the migration and installation process of CICS. Now, program processing and program control resource definitions must be defined and reside in the CSD. Therefore, you must have a CSD defined in your CICS/ESA system, with all the associated programs and transaction resources.
- The obsolete DFHPCT and DFHPPT macros are not shipped with CICS Transaction Server. However, you are recommended not to migrate the PPT entries to the CSD, but use the new autoinstall facility for programs and MAPS instead.
- CICS no longer supports the use of the file control table for VSAM object files, data tables, or shared resource pools. Resource definitions for these VSAM objects can be defined in, and installed from, the CICS system definition (CSD) data set only. CICS/ESA installs only BDAM file definitions from the FCT.

6.1.6 CICS Macro Resource Definition Table Changes

Below are commonly identifiable changes required to migrate a CICS/VSE system to CICS TS macro resource definition. These parameters listed below should be viewed as a reminder of items to consider, and not as an inclusive list of parameter changes and/or obsolescence. You should review the *CICS Macro Definition*, SC33-1648 manual for full details of parameters required for the different macro resources and the *CICS System Definition Guide*, SC33-1682 for the System Initialization parameters.

ALT is obsolete, restructuring of CICS eliminates application load table.

DCT Migrate DCT entries to the CSD. It is imperative to use the new DCT supplied definitions. There are some new entries, that if not chosen could cause the disappearance of messages. One such queue is CDUL, where dump information is written.

Also, you should remove the parameters such as REUSE, RSL, and DEVICE from the DCT specification; these parameter are obsolete.

Note: CICS TS provides a facility that allows an extrapartition data set to be used to submit jobs to MVS.

FCT Remove the CSD entry from FCT (it is now in the SIT), plus all VSAM entries (VSAM resource entries are autoinstall) from the FCT, then re-assemble the FCT and migrate the table to the CSD.

JCT ″INPUT″ JOUROPT and BUFSUV parameters are obsolete. CICS Transaction Server does not support tape logging. Also, note that log format is SMF only.

MCT Additional control over what is monitored is available with CICS/ESA. Measuring CPU time is no longer an option, CICS always measures CPU time. Also, parameter CONV is replaced by SIT parameter MNCONV. If your existing MCT specifies CONV=YES, you should remove this and specify MNCONV as a system initialization parameter (or you can set the option dynamically using a CEMT SET MONITOR or EXEC CICS SET MONITOR command).

NLT is obsolete.

PCT is obsolete, you must use RDO. Be sure to migrate all PCT entries to CSD on VSE before migrating to CICS/ESA.

PLT The sequence of events during initialization is changed in CICS/ESA. In particular, there are now two phases of program list table (PLT) processing during initialization. These two phases are separated in the same way as the first and second quiesce PLT shutdown programs, by the inclusion of DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM at the appropriate point among the DFHPLT TYPE=ENTRY macros for the programs. During the first phase, you can run only the programs that enable your user exits for restart processing. Another result of the change to PLT processing during startup is in the way CICS loads the PLT. This change means that you no longer need an RDO entry for the PLT itself. However, you must continue to define to CICS the PLT programs listed in the table by suitable program entries in the CSD. Please note that the first-phase PLTPI programs must run in AMODE 31 mode, otherwise they fail with an ABEND 0C4.

PPT is obsolete, you must use RDO. You are recommended not to migrate the definitions, but use the new autoinstall facility for programs and maps instead. Plus, you should review all resident programs on your CICS/VSE system. If the only reason a program was made resident was due to high usage, this reason is now eliminated (CICS/ESA uses an LRU algorithm for compression).

SNT DFHSNT macro is obsolete and no longer supplied. To define user attributes you must add them to the user profiles maintained by your external security manager. If you are using RACF, you define user attributes in the CICS segment of the user profiles in the RACF database.

SRT review the default CICS/ESA table for additional entries, other than the standard MVS error codes.

TCT review the entire table, particularly for BTAM changes. Add CONSLID=console designation for MVS Multiple Console Support (MCS). You must remove any spool parameter associated with CICS/VSE Report Controller Feature (RCF).

SIT review all the entries since there are VSE and MVS only parameters. Plus, there are new and different suffixes for the pregenerated CICS/MVS management modules. For example:

ABDUMP is removed because of CICS reconstructed dump facilities.

AIEXIT provides the user-replaceable autoinstall program name (was the second value for CICS/VSE autoinstall SIT parameter).

AILDELAY is the delete delay period for autoinst (was the fourth value for CICS/VSE autoinstall SIT parameter).

AIQMAX maximum number of term queued for autoinst (was the first value for CICS/VSE autoinstall SIT parameter).

AKPFREQ this parameter now represents the number of write operations to the log stream buffer before an activity keypoint is taken. The default value is now 4000 instead of 200.

AMXT is obsolete, with the new CICS dispatcher algorithms which removed the need to limit the number of active tasks.

AUTINST is replaced by the AIEXIT parameter.

COBOL2 is obsolete, CICS will initialize it immediately if the COBOL2 library is available.

CMXT is replaced by the MAXACTIVE parameter that is provided on the new TRANCLASS resource definition.

DBP is obsolete, and all backout is now coordinated by the recovery manager.

DBUFSZ is obsolete--all CICS system log output is written directly to the system log stream.

DCT the COLD option is removed, and the default is changed from YES to NO.

DLI is obsolete for both local and remote DL/I support.

FERS is obsolete.

ICVR 0 is the default if RUNAWAY(SYSTEM). You may have found it necessary to specify an artificially high ICVR value, to allow processor intensive transactions to run without being abended as runaway tasks. In CICS/ESA you can specify individual runaway timeout values on the transaction resource definition. This means that you can lower your ICVR value to a realistic limit for the average transactions, and have the definition for these reference the global ICVR limit by means of the RUNAWAY(SYSTEM) attribute.

ICVS is obsolete.

JCT The CICS log manager does not support journal data sets, making the journal control table obsolete. The CICS system log and journals are mapped to MVS system logger log streams (or, for some journals, SMF data sets) by means of JOURNALMODEL resource definitions.

JSTATUS CICS log manager does not support journal data sets, on either disk or tape, making this initialization parameter obsolete.

PLI removed because DOS PL/I is not supported.

SCS removed because the storage cushion size is determined by CICS.

System initialization modifications (SIMODs) are obsolete.

START the INITIAL option is added to indicate that CICS is to initialize as if this is a first-time start of the CICS region. Unlike a normal cold start, the cold start resulting from START=INITIAL causes CICS to purge the system log as well as the catalog data sets. An INITIAL start is the same as if you start CICS with a new system and newly defined catalog data sets.

TLT is replaced by the CSD function.

TRACE is replaced by new trace parameters.

TSMGSET there is no need for dynamic storage for temporary storage pointers as a result of the restructure of the temporary storage domain.

ZCP is not modifiable, thus the parameter is removed.

Note: CICS/ESA provides a default source (DFHSIT$$) with which you may start or use the default load module (DFHSIT) and override the defaults.

6.1.7 CSD and RDO Considerations

Below are commonly identifiable changes required to migrate a CICS/VSE system to CICS TS CSD and online resource definition entries. These parameters listed below should be viewed as a reminder of items to consider, and not as an inclusive list of parameter changes and/or obsolescence. You should review the *CICS Resource Definition Guide*, SC33-1684 for full details of parameters required for the different online resource definitions.

6.1.7.1 CSD

To start with the CSD is mandatory in CICS/MVS, but does not require an entry in the FCT.

You should define and initialize the CSD from the CICS/ESA system, as opposed to attempting to upgrade the CICS/VSE CSD to CICS/ESA. The reason is you want the space allocation of your CSD to accommodate the addition of VSAM FCT and DCT entries plus entries for comment fields (that is, entries not available to CICS/VSE). After the CSD is defined and initialized on the CICS/MVS system, you can use the DFHCSDUP utility to EXTRACT your user definition from CICS/VSE, then import it to CICS/MVS′s CSD. For details on using DFHCSDUP refer to your CICS Resource Definition Online and CICS Customization Guide.

Warning: When you migrate your CSD entries you must ensure that you do **not** copy over IBM supplied definitions in groups that you have defined. The reason is that some of the groups and resources were changed and/or deleted. Thus, you should see that your user groups with IBM defined resources are copied from the newly defined CSD. Once, you have imported your defined resources into the CSD for MVS, be cognizant of changes via the ALTER command, the default attributes and/or new attributes may not be what you desire. To illustrate, the SESSION resource definitions allows the send and receive prefixes to default. CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These three-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SENDCOUNT or RECEIVECOUNT value.

6.1.7.2 RDO

Here are items to consider with the use of RDO on your CICS/ESA system:

- Be sure that all user defined TRANSACTION resources specify the attribute (SPURGE). SPURGE must be set to purge transaction from the system and to detect loops.
- CICS-supplied transactions have changed (for example, CSSN and CSSF are CESN/CESF). CICS TS removed transactions CSMT, CSOT, CSSF, CSSN, and CSST. Please refer to your *CICS Supplied Transaction*, SC33-1686 for more details.

Below are examples of resources and/or attributes changes:

TRANSACTION/TCLASS
: is replaced by the TRANCLASS parameter, within the new TRANCLASS resource definition.

CONNECTION/SECURITYNAME(MRO)
: is obsolete on MRO connections. To specify bind-time and link security for MRO connections, you must define appropriate RACF (or another ESM) security profiles.

CONNECTION/PROTOCOL
: the scope of this parameter is extended for the external CICS interface (EXCI). This parameter allows client programs (batch programs) the ability to access CICS services.

PROGRAM/EXECKEY
: the effect of this parameter is extended for transaction isolation. With transaction isolation active, a user-key program has read and write access to the user-key task-lifetime storage of its own task only, and to any shared DSA storage, if its transaction is defined with ISOLATE(YES).

SESSIONS/RECOVOPTION
: this parameter is extended to cover VTAM persistent sessions. In earlier releases of CICS it was meaningful for CICS regions running with XRF only.

SESSIONS/RECEIVEPFX/SENDFX
: you no longer need to specify send and receive prefixes on MRO session definitions.

TRANSACTION/RESTART
this option now governs restart in two separate types of situation.

TYPETERM/RECOVNOTIFY
the scope of this parameter is extended to cover VTAM persistent sessions. In CICS/VSE this parameter was meaningful for CICS regions running with XRF only.

6.1.8 CICS System Data Sets Requirements

Before you install your CICS data sets you should determine the DASD requirements for all CICS data sets and MVS data used by CICS.

Below are the data sets needed to implement CICS TS. You should review the *CICS System Definition Guide*, SC33-1682 and the *CICS Installation Guide*, GC33-1691 for full details on the facilities, functions and usage for each data set listed below:

Temporary storage data set
Transient data destination data sets
CICS log streams
System definition data set
Catalog data sets
Auxiliary trace data sets
Dump data sets
Availability manager data sets
CMAC messages data set

After you have installed CICS, and applied any necessary service, you can run the DFHCOMDS, DFHDEFDS, and DFHCMACI jobs to create the CICS data sets. MVS storage management facility (SMS) should facilitate the placement, and allocation of your data sets. Still, you should keep an eye on extent allocations for the data sets used by CICS, for performance reasons (that is, is the single extent allocation too small).

Be sure you note that CICS Log is unique to CICS TS, and requires special consideration as opposed to journal files used in CICS/VSE. In OS/390 Version 2 Release 4, the CICS log manager supports the DASD-only option of the MVS system logger. This means that individual CICS log streams can use either Coupling Facility structures or DASD-only logging.

Figure 13 on page 146 shows the choices you have when defining individual log streams, depending on the hardware and software you are using.

Coupling facility?	OS/390 Version 2.4?	Log stream possibilities
Yes	No	All must use CF.
No	Yes	All must use DASD-only.
Yes	Yes	Individual log streams can use either CF or DASD-only.
No	No	CICS TS Release 2 not supported.
Note: Without a coupling facility, you cannot share general log streams across MVS images.		

Figure 13. Log Stream Choices Resulting from Hardware and Software Used

CICS system programmers need to consult with their MVS system programmers to plan the storage required for the log streams needed by the CICS log manager.

Figure 14 shows MVS data sets used by CICS.

SDUMP data sets	MVS SDUMP macro	Used by CICS for system dumps via the MVS SDUMP macro.
SMF data sets	System management facility	Used by CICS monitoring and statistics domains for monitoring and statistics records.
GTF data sets	Generalized trace facility	Used by CICS trace domain for CICS trace entries.

Figure 14. MVS Data Sets used by CICS

Please refer to the *OS/390 Initialization and Tuning Guide* for information on calculating the size of SDUMP. For background information about SMF and data set considerations consult the *OS/390 MVS System Management Facilities*. However, you must reference the *CICS Customization Guide* for information on CICS monitor records and sizes and the *CICS Performance Guide* for information on CICS statistics records and sizes. If you are to use CICS with GTF please review the *OS/390 MVS Diagnosis Tools and Service Aids*.

6.1.9 CICS System Program Interface and Exits

6.1.9.1 System Programming Commands

CICS system programming interface (SPI) commands provide you with the ability to access and modify CICS system information. SPI should be considered for CICS applications that presently modify and/or access CICS internal blocks. Although, SPI commands will not grant access to all CICS blocks and addresses, SPI commands either retrieve information about CICS resources or system elements from:

INQUIRE commands
COLLECT STATISTICS

or commands that modify the status or definition of the system or a resource, or invoke a system process:

SET commands
CREATE commands
DISCARD commands
PERFORM commands
ACQUIRE TERMINAL

or commands that modify or expand system execution by means of exits:

DISABLE PROGRAM
ENABLE PROGRAM
EXTRACT EXIT
RESYNC ENTRYNAME

CICS TS requires that all SPI commands specify an SP translator option and security checking for transactions issuing SPI commands. Therefore, you should review all SPI commands of CICS TS to determine what modifications and reassemblies are required to your present SPI programs. Please refer to *CICS System Programming Reference*, SC33-1689 for more details on SPI command changes and the *CICS Application Programming Guide*, SC33-1687 for information on the translator options.

6.1.9.2 Exits

Exits will require special attention and a significant amount of your conversion work effort. All exits will require a rewrite.

CICS TS does not support changes to internal control blocks. The user exit programming interface provides global user exit programs with access to some CICS services. It consists of a set of macro function calls that you can use in your user exit programs. It provides opportunities to extend CICS functions beyond the facilities provided in the standard CICS system, but it should be used with care. Any exit programs you write that use the interface must be written following the specific guidance documented in the *CICS Customization Guide*, SC33-1683, and you should carefully test to ensure that they cannot cause system errors.

The user exit programs must be in Assembler; the XPI is not provided for other languages. You should also note that programs containing XPI calls must be written to 31-bit standards, and must be reentrant.

Your program must be in primary-space translation mode when you invoke the XPI. (For information about translation modes, see the *IBM ESA/370 Principles of Operation* manual.)

Notes:

1. You cannot use all of these calls at every global user exit point. You will find an indication of when these calls cannot be used both with the description of each function call, and in the list of exit points in the *CICS Customization Guide*.

   Warning: These XPI calls are used to invoke CICS services; using them in the wrong exits causes unpredictable errors in your CICS system.

2. There is a restriction on using the XPI early during initialization. Do not start exit programs that use the XPI functions INQUIRE_MONITOR_DATA, MONITOR, TRANSACTION_DUMP, and WRITE_JOURNAL_DATA until the second phase of the PLTPI. For further information about the PLTPI, refer to ″Writing initialization and shutdown programs″ in the *CICS Customization Guide*, SC33-1683.

3. These XPI functions are likely to cause the task executing the user exit program to lose control to another task while the XPI function is being executed. Therefore, the use of XPI functions must be very carefully considered as interrupting the flow of CICS functions could cause problems, such as lockouts, to occur.

For more information on tailoring CICS global and user exits review your *CICS Customization Guide*, SC33-1683.

Remember that CICS/ESA does not support macro-level programs. If you attempt to invoke programs with macro-level code and/or internal CICS addresses the following should result:

- CICS issues a warning message naming the offending program or transaction
- CICS names the offending program or transaction and abends the offender
- CICS names the offending program or transaction and disables

Changes to the exit programming interface means that you will also need to make changes to global user exit programs. Still, you must reassemble all global exit programs. Plus, there are the following changes to be noted in your exit programs.

CSA and TCA addresses are withdrawn from DFHUEPAR.

The following API commands are not supported in global user exits:

Command	Exit points
EXEC CICS ABEND	All exits
EXEC CICS RETURN	All exits
All EXEC DLI	XRMIIN and XRMIOUT
All EXEC SQL	All exits
All CALL DLI	All exits

You must rewrite all user-replaceable modules except for DFHACEE, DFHUAKP, DFHXSP and DFHXSE user-replaceable modules, which are obsolete. Also, the DFHNTRY is replaced by a new user-replaceable module DFHREST.

Note: VSE/ESA System Package (SP) supplied a number of user exits and user replaceable modules, that are part of the packaging of VSE/ESA. As such these programs may be similar to other CICS supplied sample program, but are unique in what they offer VSE/ESA users. If you were using any of the programs below, you may want to convert the code and/or find similar solutions through IBM packages and/or vendor programs.

IESZATDX - auto install program
IESZNEP - VTAM network error program
DFHXSE and IESEXIT1 - signon program
DFHPEP - program error program (invokes OLPD transaction for ABEND)
SKEXITDA - captures VSE/ESA system activity data from the II and stores the resulting data in CICS/VSE temporary storage queues.

Note: The above programs are located in VSE/ESA ICCF library 59.

6.1.10 CICS Transaction Security

CICS/ESA security is provided through external security (that is, RACF). Hence CICS/VSE internal security needs to be converted to an external security facility.

In the MVS environment, RACF provides an external security manager. RACF controls access to data sets from CICS, TSO, and batch.

The recommendation is to migrate to RACF and CICS/ESA external security.

If you are using RACF as the external manager, consider:

- All CICS started task names must be defined as user IDs having the authority to execute all transactions *UACC(READ)*.
- All transactions must be defined to RACF (even previously unsecured transactions).
- If using transient data initiated transactions or transactions started on a terminal, you may need to add an XPCT profile, or allow the default user UACC(READ).

New CICS command RACF resources: EXITPROGRAM, REQID, and STORAGE, update authority is required to enable, disable, extract, or resync EXITPROGRAM, and may be administered from the PLT process.

If you are using CICS for VSE/ESA 2.3, you can use the security migration aid to assist you with your migration of your CICS internal security definitions to an environment where the resources can be defined with RACF. You will need the CICS/VSE Security Migration Aid (supported via APAR PN87442) and the *CICS/VSE Security Migration Aid*, SC33-1406 manual.

6.1.11 CICS UPSI

There is no UPSI in MVS. Execution overrides are in the PARM field of the JCL statement - EXEC PGM=DFHSIP. The following list identifies the CICS/MVS equivalents:

Bit 0	SYSIPT overrides yes or no.	All overrides are passed as execution parameters to DFHSIP.
Bit 1	This is no longer required for CICS/OS or CICS/VSE 1.6 or later.	
Bit 2	Operator prompt for overrides yes or no.	The parameter 'CONSOLE' as the last PARM in the EXEC statement indicates that the operator is to be requested to enter overrides.
Bit 3	If Dump returns a nonzero return code when writing to the SYSDUMP data set and an IDUMP will be produced on SYSLST.	CICS/MVS DUMP is directed to MVS SYS1.DUMxx only.
Bits 4-5	Are not currently used.	
Bits 6-7	Reserved for DL/I.	Not required for IMS.

6.1.12 Application Programming

Command-level programs are upward compatible at both source and object level, provided they conform to the interface as defined in the Application Programmer's Reference manual. However, it is imperative that you understand upward compatible does not constitute that your program will continue to run on your CICS/VSE system (that is, programs reassembled and linkedited to CICS TS are not guaranteed to run on CICS/VSE). Hence, you should be sure that your System Management/Change Management and Roll-back plan accounts for this situation.

Notes:

1. CICS 1.7 applications can be relinkedited on MVS, but you should not expect the programs to function (consider recompiling programs).
2. DOS PL/I applications do not function on MVS, hence you should consider rewriting DOS PL/I to a PL/I supported level.

Macro-level programs are not supported on MVS. You can convert your macro-level programs to command-level using the CICS AMA conversion aid to assist with the conversion.

CICS programs written with LE support are generally object module compatible between VSE/ESA and OS/390. However, there are some commands where CVDA/REP values are different, thus retranslation, recompile is necessary. CICS command-level (non-LE) programs which have adhered to the CICS documented Application Programming Interfaces (API) are generally source compatible between VSE/ESA and OS/390.

Source programming conversion should be minimal with only language differences to be resolved, such as COBOL reserved words. The presence of any of the following conditions may substantially increase the effort:

- Programs that start a BROWSE operation, then read for UPDATE. For a file accessed in non-RLS mode, CICS should return an INVREQ condition. Updating and deleting records in a browse is only supported for VSAM files opened in RLS mode.
- EXEC CICS ADDRESS CSA commands are no longer supported.

- RPG II is not a supported language for CICS/OS. RPG II programs should be converted to a supported application language (that is, COBOL, PL/I, C++, and or Assembler).
- Programs that directly invoke operating system services.
- Programs that directly access operating system control blocks.
- Programs that access internal CICS control blocks (DSECTs).
- The CICS/VSE Report Controller Feature (RCF) is not supported by CICS/MVS. This includes many suboperands of the EXEC CICS SPOOL commands. The basic spool interface (open, close, read, and write) functions are available in both CICS/VSE and CICS/MVS.
- The CICS provided programming interface to JES (the Job Entry Subsystem component of MVS) allows CICS applications to create and retrieve spool files. Spool files are managed by JES and are used to buffer output directed to low-speed peripheral devices (printers, punches, and plotters) between the job that creates them and actual processing by the device. Input files from card readers are also spool files and serve as buffers between the device and the jobs that use the data.

  The interface consists of five commands:

  SPOOLOPEN INPUT, which opens a file for input
  SPOOLOPEN OUTPUT, which opens a file for output
  SPOOLREAD, which retrieves the next record from an input file
  SPOOLWRITE, which adds one record to an output file
  SPOOLCLOSE, which closes the file and releases it for subsequent processing by JES.

  Spool Interface restrictions

  There are internal limits in JES that you should consider when you are designing applications. Some apply to JES2, some to JES3 and some to both. In particular:

  JES2 imposes an upper limit on the total number of spool files that a single job (such as CICS) can create. If CICS exceeds this limit during its execution, subsequent SPOOLOPEN OUTPUT commands fail with the error condition.

  JES3 does not impose such a limit explicitly, but for both JES2 and JES3, some control information for each file created persists for the entire execution of CICS. For this reason, creating very large numbers of spool files can stress JES resources.

  Spool files require other resources (buffers, queue elements, disk space) until they are processed. Please review the *CICS Application Programming Guide* for more details.

  However, spool read is single thread in CICS/MVS. This may have significant performance implications. Similar functions may be provided in the MVS environment by the Report Management and Distribution System (RMDS), 5665-310.

- If your CICS applications have exploited the menu services provided by the VSE Interactive Interface (II) they may need some rework. The II selection panels and II programs such as IESFPIP do not exist in CICS/ESA. The functions may be provided by user written CICS programs and maps. Similar

functions are provided by the Application Support Facility for MVS (ASF), 5685-043.

Some CICS programs written in assembler language may have to be reworked. These applications are more prone to violate the CICS API restrictions.

Also, be sure to review all EXEC CICS commands for changes in VALUES.

The name of the CSECT within module DFHECI changed from DFHECI to DFHELII. So, be sure that your LINKEDIT included DFHECI.

An application program that passes the address of a COMMAREA to another application program can be above 16MB, below 16MB, or it can be a zero address. A COMMAREA can be in CICS-key storage or USER-key storage (if CICS is running with storage protection), or in read-only storage (possibly obtained using an MVS GETMAIN call). The length of the COMMAREA can be a positive value or zero, but a negative value always results in an error. Therefore, the user must provide a condition check for a negative value in the user programs.

Any Assembler programs which use the DFHEISTG copybook should be reviewed. It increased by 136 bytes.

If you have any Distributed Transaction Processing (DTP) applications that use CICS APPC commands, changes to the CICS implementation of the APPC architecture could mean you need to change the APPC applications before you can migrate them to CICS TS.

Please review the *CICS/ESA Distributed Transaction Programming Guide*, SC33-1174 for more details.

CICS supports the following Assembler, COBOL, PL/I, and C/370 compilers:

- High Level Assembler/MVS & VM & VSE Version 1.1 (5696-234)
- IBM PL/I for MVS & VM (5688-235)
- OS PL/I Optimizing Compiler Version 2 Release 1 (5668-910)
- OS PL/I Optimizing Compiler Version 1 Release 5.1 (5734-PL1), or later
- IBM COBOL for MVS & VM (5688-197)
- VS COBOL II (5668-958 and 5688-023)
- IBM C/C++ for MVS/ESA (5655-121)
- C/370 (5688-040 and 5688-187).

CICS also supports IBM Language Environment for MVS run-time environment (5688-198), with the following SAA AD/Cycle COBOL, C/370, and PL/1 SAA AD/Cycle compilers:

- *SAA AD/Cycle COBOL/370 (5688-197)*
- *SAA AD/Cycle C/370 (5688-216)*
- *SAA AD/Cycle PL/I (5688-235).*

Below are recommended conversion aids available to assist you with the conversion of your CICS application programs.

The DFHMSCAN utility program, which is available with CICS/VSE is recommended for reviewing CICS application program libraries. This program can be run against VSE application libraries to find out which application

programs use CICS macros, which is very useful when you must determine your scope-of-effort.

The CICS Application Migration Aid (5695-061), should be used to assist customers migrating macro-level programs to command-level programs. Please review the manual *CICS/VSE Application Migration Aid Guide V2*, SC33-1901 for more detail.

COBOL and CICS Command Level Conversion Aid for VSE (CCCA) - CCCA assists in the removal of BLL cell processing to ANSI 85 COBOL processing (5785-CCC). Please review the manual *CCCA/VSE Installation and User's Guide*, SC26-8269 for more details.

During the planning and/or application conversion process you may find it difficult to convert your macro programs. You may find other aids to help you with the conversion and/or consider the coexistence of CICS/VSE and CICS TS systems.

6.1.13 CICS/VSE and TS Coexistence Considerations

As part of the migration you may need to consider the coexistence of a CICS/VSE system with a CICS TS system via an ISC connection. The reasons for this may vary from parallel testing, migrating of data, to function shipping requirements for DL/I and so on. Still, you should understand functions that will help the cooperative systems.

If you allow the send and receive prefixes to default, CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These three-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SEND- or RECEIVECOUNT value. This method is the same as that for APPC sessions.

To maintain compatibility with earlier releases, this change is optional. You can continue to define your own prefixes for the send and receive sessions, in which case CICS generates the terminal control table entries (TCTTEs) for session names in the same way as for earlier releases.

Please review the *CICS Intercommunication Guide*, SC33-1698 which provides definitions and guidelines for intersystems connections.

6.1.14 Testing and Problem Determination Considerations

You should consider that due to the change in operating systems and CICS′s preparation, education and training should be completed before your installation and test begin. However, here are few items to consider with the process.

There are major differences in:

- initialization messages.
- system initialization parameters.
- initialization error recovery.
- messages and codes issued by different systems, and in the operator actions they require.
- handling output from CICS monitoring.

- handling output from CICS dumps.
- handling output from CICS trace.
- handling output from CICS statistics.
- problem determination.
- restart and recovery requirements.
- security administration.
- application of software services.

Identify and understand the different IBM and vendors support structures and procedures. You should have available:

personal names of your contact points
telephone numbers

Therefore, you should see that your system management procedures are updated. The following manuals *CICS Operations and Utilities Guide*, SC33-1167, *CICS User's Handbook*, SX33-6104, *CICS Messages and Codes*, GC33-1694, *CICS Glossary* GC33-1705, and *CICS Problem Determination Guide*, GC33-1693 should be used during these periods.

6.1.15 Vendor Applications

In CICS Transaction Server for OS/390, the autoinstall user program invoked for installation and deletion of virtual terminals is used by the External Presentation Interface (EPI) and terminal emulator functions of the CICS Clients products. They are defined to CICS as remote 3270 devices. You should be sure your vendor products will work with the supported autoinstall program.

For an introduction to the CICS Clients products, and detailed information about OS/390 support for them, see the *CICS/ESA Server Support for CICS Clients* manual.

Customers should be advised to contact the suppliers of any third-party software used with CICS to ensure that the supplied packages will run with CICS Transaction Server for OS/390.

6.2 CICS with DL/I

The CICS TS - IMS/VS interface is implemented differently than the CICS/VSE - DL/I interface.

If you are a CICS local DL/I user you must plan to migrate your databases to DBCTL. Alternatively, you can use CICS function shipping to CICS/VSE DL/I data sets. These are the only two methods of DL/I database access that CICS continues to support.

For information about migrating to DBCTL see the *CICS IMS Database Control Guide*.

Chapter 7. ICCF and TSO

DOS/VSE users of the Interactive Computing and Control Facility (ICCF) who migrate to OS/390 will find a very powerful interactive system available via OS/390′s Time Sharing Option (TSO/E) and related products, particularly the Interactive System Productivity Facility (ISPF). This section addresses the TSO/E and ISPF implementation of the common functions used in ICCF. It is not intended to be a complete list of the functions available to the TSO/E user.

7.1 Preparing to Use the System

ICCF uses a single direct access data set, DTSFILE, to maintain the information and data necessary for interactive execution. DTSFILE is logically divided to contain user profiles, ICCF libraries, and interactive input, list, and punch areas. TSO/E, on the other hand, maintains user profile information in either the Resource Access Control Facility (RACF, or OS/390 Security Server) database or (less commonly) in the TSO/E User Attribute Data Set (UADS). In addition, you can tailor the interactive user′s environment by assigning customized LOGON procedures stored in a partitioned data set (PDS). The equivalent of an ICCF library would be either a sequential data set or a PDS in the TSO/E environment. TSO/E interfaces directly with the job entry subsystem, JES2 or JES3, to handle interactive job input and list or punch output. In this section we will review the requirements to allow access to your TSO/E system.

7.1.1 User Profiles

The ICCF System Administrator authorizes ICCF users by creating a user profile and storing it in DTSFILE. The person responsible for TSO/E in an MVS environment will normally authorize TSO/E access by creating user profiles in the RACF data base and defining a TSO ″segment″ for those users. Alternatively, though much less common since most OS/390 systems have RACF active, the administrator could create entries in the TSO/E User Attribute Data Set (UADS). This chapter will focus on using RACF to register TSO users. If you need information on using the older TSO/E methods you can read about the ACCOUNT command in the *TSO/E Customization* book.

As delivered, your OS/390 system with RACF will have one user defined, IBMUSER. IBMUSER has a predefined password of SYS1, which you must change the first time you logon, and has the SPECIAL attribute to allow full administration of RACF. As one of your first tasks, you would create another administrative ID and then ″revoke″ IBMUSER to make it unusable by any other users who might attempt to take control of your system.

Each TSO/E user has, as a minimum, a user ID and an associated LOGON procedure. LOGON procedures will be covered in the next section. The user ID can be from 1-7 alphameric characters beginning with an alphabetic or a national character. An ICCF user ID is always four characters.

Each user will also have a password, which you assign initially when creating the user′s profile. RACF will mark this password as expired and require the user to change it upon first logon. Each user may change the password periodically if he or she desires, and through RACF′s options you may enforce such periodic changes. For more information on this please refer to the *OS/390 Security Server (RACF) Security Administrator′s Guide*.

You may choose to assign account numbers to your users for accounting or other purposes. This account number can be from 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character. You use the RACF ACCTNUM resource class to authorize use of account numbers. Please refer to the *TSO/E Customization* book or the *RACF Security Administrator's Guide* for more details on account numbers.

TSO/E allows you to specify the authority to use or a restriction against using the ACCOUNT, OPERATOR, SUBMIT, STATUS, CANCEL, and OUTPUT commands by defining resource profiles in RACF's TSOAUTH resource class. Again, *TSO/E Customization* and the *RACF Security Administrator's Guide* have more information on this topic.

You use commands similar to the following to create a TSO/E user with roughly the capabilities of the ICCF System Administrator. You issue the RDEFINE command only once, and for subsequent users you add you do not need the RDEFINE.

```
ADDUSER AAAA PASSWORD(secret) SPECIAL
ALTUSER AAAA TSO(PROC(LOGROUT))
RDEFINE TSOAUTH (ACCT JCL OPER MOUNT PARMLIB) UACC(NONE)
PERMIT ACCT    CLASS(TSOAUTH) ID(AAAA) ACCESS(READ)
PERMIT JCL     CLASS(TSOAUTH) ID(AAAA) ACCESS(READ)
PERMIT OPER    CLASS(TSOAUTH) ID(AAAA) ACCESS(READ)
PERMIT MOUNT   CLASS(TSOAUTH) ID(AAAA) ACCESS(READ)
PERMIT PARMLIB CLASS(TSOAUTH) ID(AAAA) ACCESS(READ)
```

Of course, AAAA will not normally need authority to use the ACCOUNT command (ACCT resource in the TSOAUTH class) but it does not hurt for AAAA to have this authority and it may prove helpful at some time. As an administrator, though, AAAA could give himself this authority. You might also wish to choose different "universal access" rules (UACC) for the JCL resource, which gives the ability to submit batch jobs. Often all users can submit batch jobs, and you would assign a UACC of READ to cover this situation.

In this example, TSO/E user AAAA with password "secret" uses a LOGON procedure named LOGROUT. He has no default account number, and TSO/E does not check authority to use account numbers until you configure the RACF ACCTNUM class. AAAA has authority to use the ACCOUNT command (ACCT), the OPERATOR command (OPER), and the SUBMIT, STATUS, CANCEL, and OUTPUT commands (JCL). He is also able to request volume mounts as necessary. In addition, AAAA has authority to tell TSO/E, via the PARMLIB command, to change its configuration parameters. TSO/E will normally use the parameters contained in member IKJTSO00 in partitioned data set SYS1.PARMLIB. After a change to this member, the TSO/E PARMLIB command will tell TSO/E to use the new parameters without requiring a system IPL.

A terminal user who will be using TSO/E for application development will also have a user profile. However, such a user would probably not have authorization to use the ACCOUNT or OPERATOR commands, nor would he be authorized to request volume mounts.

The TSO/E Information Center Facility (ICF) provides an ENROLL facility for the TSO/E administrator. This facility will add TSO/E users to RACF or UADS (the administrator's choice) as well as performing other necessary tasks.

7.1.2 LOGON Procedures

In ICCF, a logon procedure may be specified in the user profile. This entry references an ICCF procedure or macro used to define the environment for this logon. These optional procedures or macros are normally defined by the user if they are present.

In TSO/E, the LOGON procedure is not optional. The LOGON procedure defines the system resources available to a terminal user and defines or allows for dynamic allocation of all data sets used by a terminal user. LOGON cataloged procedures must reside in the data set defined in the procedure used to start the primary job entry subsystem, JES2 or JES3. This data set may be either SYS1.PROCLIB or a partitioned data set dedicated to LOGON procedures.

You may specify a user's default logon procedure (for the user's first logon) in the user's TSO segment using the PROC keyword. You may authorize or restrict usage of logon procedures using RACF's TSOPROC resource class. Again, see *TSO/E Customization* and *RACF Security Administrator's Guide* when you need more details.

7.1.3 Message Facilities

The ICCF member A$MAIL normally resides in the ICCF common library of DTSFILE and is used to broadcast messages to all ICCF users. The ICCF command /MAIL is issued by an ICCF user to view any messages that have been stored in member A$MAIL. If messages are sent to an individual ICCF user by using the /SEND command, they are stored in an ICCF member unique to the receiver that is created automatically by ICCF. Both of these ICCF facilities are optional.

For the TSO/E environment, a Broadcast Data Set, SYS1.BRODCAST, is required. Normally, though, you will use the broadcast data set only to hold notices, messages intended for display to all users at logon time such as a message of the day or a system status message. For messages directed to individual users (single-line mail) you will normally want to configure TSO/E to use a separate data set for each user. You do this using operands on the SEND statement in SYS1.PARMLIB(IKJTSO00). Smaller installations may wish to use SYS1.BRODCAST for mail messages, too, and can configure this using the SEND options in IKJTSO00 if they desire.

TSO/E users can choose to view mail and notices at logon time, or to suppress such viewing by specifying NONOTICES and/or NOMAIL. They may also view mail and notices whenever they desire using TSO/E's LISTBC command.

7.1.4 Security

ICCF provides facilities which protect ICCF libraries, ICCF library members, files, and VSE library members against unauthorized access from interactive partitions. The implementation of security in the ICCF environment is not related to an overall DOS/VSE security implementation.

In the MVS TSO/E environment, security is an MVS system level requirement and will normally be handled through RACF.

Both ICCF and TSO/E provide a first level of security in the requirement for predefined user IDs before accessing the system. A password for the user ID is required for access to the system.

ICCF provides another level of security by defining ICCF libraries within DTSFILE as either PUBLIC, PRIVATE, or COMMON. All ICCF users have read access to data stored in the single COMMON library supported by ICCF. However, only ICCF users with a System Administrator level profile have write access to this library. Multiple PUBLIC ICCF libraries are supported in DTSFILE and are normally used to store data that can be read by any ICCF user, but updated only by the originator. ICCF PRIVATE libraries are normally used to store data that can be accessed by users authorized for access to that library.

With RACF you can specify system options (via the SETROPTS command) which tell RACF how to protect data sets, and in particular whether to allow access to unprotected data sets or not. If you choose to require protection for all data sets (SETROPTS PROTECTALL) then you will have to define DATASET profiles before anyone can access data sets. (Obviously you would want to create such profiles before you specify PROTECTALL.) If you don't enforce protection of all data sets, then you can identify those data sets which do require protection and define DATASET profiles just to protect them. The *RACF Security Administrator's Guide* has information on protecting resources, both data sets and other kinds, using the ADDSD, RDEFINE, and PERMIT commands.

In the TSO/E environment, you can use RACF to restrict or allow access to a PDS to simulate the library access defined above. The TSO/E equivalent of the ICCF COMMON library is a PDS with a universal access level of READ and an access list with only a few users having UPDATE authority. Since TSO/E command lists (CLISTs) and REXX execs, equivalent to ICCF procedures, are stored in a PDS, you might define a single CLIST PDS for storing all common CLISTs available to any TSO/E user. This PDS is similar in use to the ICCF PUBLIC library. The TSO/E equivalent of an ICCF PUBLIC library is a PDS with, again, a universal access of READ and an access list with a limited number of users with UPDATE authority. For an ICCF PRIVATE library equivalent PDS under TSO/E, you specify a universal access level of NONE and then permit the necessary users with either READ or UPDATE authority, as appropriate, via the access list of a DATASET profile.

Since protection is at the data set level in TSO/E, it is not possible to do member level protection.

7.1.5 Summary

Although you can begin using TSO/E with a minimum amount of knowledge in the areas of User Profiles and LOGON Procedures, there are many options available in preparing TSO/E for your interactive users. You should review *TSO/E Customization* for details on these subjects. Security is a very important aspect of your new MVS system and should be reviewed at the system level not just for your TSO/E system. For information on the OS/390 Security Server (RACF) you can begin with the *RACF General Information* manual, though administrators will also need to study the *RACF Security Administrator's Guide*.

7.2 Using the System

Once a TSO/E user has access to his new interactive system, he will need to know how he can accomplish what he used to do with ICCF. In this section we will explain how to implement ICCF functions in a TSO/E environment.

7.2.1 Accessing the System

Since LOGON to TSO/E is dependent on the telecommunications access method used with TSO/E, the System Standards implemented by the Systems Programmer, and the related program products installed, you should reference the *TSO/E Primer* and your Systems Programmer for information on logging on to TSO/E.

7.2.2 Entering and Manipulating Data

In ICCF, data is entered and stored as a member of an ICCF library. Data is restricted to 80 byte records in an ICCF library. You may enter data into an ICCF library member from Input, Edit, or Full Screen Edit mode. ICCF allows modification of data stored in members of ICCF libraries only. Modifications are made while in Edit or Full Screen Edit modes and physically change the data on the DTSFILE.

In TSO/E, data is entered and stored in sequential data sets, or partitioned data sets (PDS) using the Interactive System Productivity Facility/Program Development Facility (ISPF/PDF) editor or the TSO/E EDIT command and its subcommands. Most users use ISPF/PDF, rather than the older TSO/E EDIT command, to gain increased usability and improve their productivity. Record formats may be either fixed or variable with a logical record size less than or equal to 255 and a block size less than or equal to track length. A PDS is a data set partitioned into one or more independent groups called members. Each member must have a unique name and can be referred to separately by appending the member name, enclosed in parentheses, to the data set name.

The name you give a data set should follow the TSO/E naming conventions. A TSO/E data set name normally has three fields.

Identification Qualifier - This is always the leftmost qualifier of the full data set name. For TSO/E, this qualifier is the prefix selected in the PROFILE command. If no prefix has been selected, your user ID will be used.

User-Supplied Name - You choose a name for the data sets that you want to identify. It can be a simple name or several simple names separated by periods.

Descriptive Qualifier - The descriptive qualifier is always the rightmost qualifier of the full data set name. It is one of a set of keywords that describe the contents of the data set to the system (that is, ′data′ identifies the data set as uppercase text). A list of standard descriptive qualifiers and the respective contents follows.

Descriptive Qualifier	Data Set Contents
ASM	Assembler input
CLIST	TSO/E commands and subcommands
CNTL	JCL and SYSIN for SUBMIT command
COBOL	COBOL statements
DATA	Uppercase text
FORT	FORTRAN statements
LINKLIST	Output listing from linkage editor
LIST	Listings
LOAD	Load module

Descriptive Qualifier	Data Set Contents
LOADLIST	Output listing from loader
OBJ	Object module
OUTLIST	Output listing from OUTPUT command
PLI	PL/I statements
TESTLIST	Output listing from TEST command
TEXT	Uppercase and lowercase text
VSEBASIC	VSBASIC statements

Each field of a data set name consists of 1-8 alphameric characters and begins with an alphabetic or national ($, @, and #) character. The fields must be separated by periods. The total length of the name, including periods, must not exceed 44 characters.

The data set naming conventions simplify the use of data set names. If the data set name conforms to the conventions, you need specify only the user-supplied name field (in most cases) when you refer to the data set. The system will add the necessary qualifiers to the beginning and to the end of the name that you specify.

For example, entering the TSO/E EDIT command

```
EDIT PAYROLL(PRTCHK) NEW COBOL
```

puts your terminal into the Input mode of EDIT on a new member (PRTCHK) of a PDS with the name USERID.PAYROLL.COBOL where the USERID is the TSO/E user ID from UADS. This does not work when using ISPF, however, but even in this case you should use the descriptive qualifiers to provide a good indication of the contents of your data sets, which will make it easier for you to work with them.

In some cases, however, the system will prompt you for a descriptive qualifier. Until you learn to anticipate these exceptions to the naming conventions, you may wish to specify both the user-supplied name and the descriptive qualifier when referring to a data set.

Using TSO/E, data sets can be created and edited by subcommands of EDIT which reference line numbers or text within lines. The first method is called line-number editing and the second, context editing. These two methods can be used interchangeably. TSO/E does not offer a full screen edit capability. Again, though, with the installation of the Interactive System Productivity Facility (ISPF) program product a full screen editor is available to the TSO/E user and most users would not choose to use the TSO/E EDIT command. In fact, most TSO/E users will spend their entire session within ISPF using only its panels or graphical user interface.

For detailed information on the TSO/E EDIT command see *TSO Extensions Command Reference*. For information on ISPF see *ISPF Getting Started*.

7.3 Executing Programs at a Terminal

Both ICCF, TSO/E, and ISPF provide commands to compile, link-edit, and execute (or compile and load) your source program at the terminal. They also allow you to use other programs, such as utilities at the terminal.

Under ICCF programs that expect input from the console will read input from your terminal. Card input is either entered from the terminal (/DATA INCON) when requested from SYSIPT or SYS005 or can be included from an ICCF library member (/DATA NOINCON followed by /INCLUDE). List output will be returned to your terminal.

Under TSO/E, you define your input and output data sets via the ALLOCATE subcommand of the EDIT command, or the ALLOCATE command. You may allocate a data set to the terminal by using an asterisk (*) as the data set name. The following example shows the use of the ALLOCATE command for allocating the data sets required for an execution of the Assembler.

```
   .
   .
READY
allocate dataset('sys1.maclib') file(syslib) shr
READY
allocate file(sysut1) new block(400) space(400,50)
READY
allocate file(sysut2) new block(400) space(400,50)
READY
allocate file(sysut3) new block(400) space(400,50)
READY
allocate dataset(*) file(sysprint)
READY
allocate file(syspunch) sysout
READY
allocate dataset(prog.obj) file(sysgo) new block(80) space(200,50)
READY
allocate dataset(input.asm) file(sysin) old
   .
   .
```

The ALLOCATE commands in the example would define the macro library to be used by the assembler (SYSLIB), the assembler work data sets (SYSUT1, SYSUT2, and SYSUT3), a data set for the punched deck of an object module (SYSPUNCH), a data set for the link and go object deck (SYSGO), the input to the assembler (SYSIN) which is a data set with the fully qualified name 'userid.INPUT.ASM', and the output of the assembler (SYSPRINT) which is to be directed back to the terminal.

Note that rather than using the commands shown above, your users will probably wish to use ISPF's facilities for invoking the assembler or compilers. These facilities, commonly available from option 4 of the main ISPF panel, automate much of the work of invoking compilers, assemblers and so on.

If you have to allocate the same data sets every time you log on, you can have your installation allocate them in the form of fully defined data sets in the LOGON procedure or you can build a command procedure containing your ALLOCATE commands and execute that procedure as soon as you are logged on.

7.4 Submitting Jobs for Batch Execution

ICCF allows users to submit jobs for batch execution through the SUBMIT procedure and an ICCF supplied program, DTSSUBMT. Tailoring of the SUBMIT procedure allows the ICCF System Administrator to provide system standards for execution and list and punch output. Listed output from the batch execution may be viewed at the terminal using the /LISTP command provided that the output is dispatchable and is not presently being printed by VSE/POWER. ICCF also provides three procedures, GETL, GETP, and GETR, to retrieve list, punch or reader queue data and store this data as ICCF members. You may use the /DQ command of ICCF to view the VSE/POWER reader, list, and punch queue directories, and the ICCF supplied program, DTSDA, can be used to display the status of the DOS/VSE partitions.

In TSO/E, you can submit jobs for batch processing if your installation authorizes you to do so. This authorization, specified as JCL, is stored in RACF or UADS with your user attributes. If you have this authorization, the system lets you use the four commands (SUBMIT, STATUS, CANCEL, and OUTPUT) that control the processing of batch jobs. You can use these commands to submit a batch job, to display the status of a batch job, to cancel a batch job, and to control the output of a batch job. You may also use ISPF facilities to perform this work rather than the commands TSO/E supplies. Many JES2 customers also use the facilities provided by the System Display and Search Facility (SDSF) to control and work with the output from batch jobs. SDSF provides a more complete full-screen interface to batch jobs, and gives functions similar to the /CTLP and /DQ facilities of ICCF.

When you enter the SUBMIT command, you must give the name of a data set (or data sets) containing the batch job (or jobs). Each job consists of job control language (JCL) statements and of program instructions and data. If you do not specify the NONOTIFY operand, you will be notified when your batch job terminates. TSO/E provides for system standards on submitted jobs through the coding of a SUBMIT command exit. Through this exit, an installation can approve, reject, or modify the JCL statements being submitted.

Any time after you submit a background job you can use the STATUS command to have its status displayed. The display will tell you whether the job is awaiting execution, is currently executing, or has executed but is still on the output queue. The display will also indicate whether a job is in hold status. The STATUS command is similar to the /STATUSP command of ICCF.

The CANCEL command cancels execution of a batch job. This command can only be used on jobs that follow the naming convention of job names beginning with the TSO/E user ID. There is no equivalent to this command in ICCF.

The OUTPUT command may be used to manipulate all held output, regardless of whether the output is produced during the current LOGON session, a previous LOGON session, or by a batch job submitted from any source.

7.4.1 Using Command Procedures

Both ICCF and TSO/E provide the capability of storing frequently executed commands or lists of commands. In ICCF these stored commands are called Procedures or Macros. They are stored as an ICCF library member. In TSO/E they are called Command Lists (CLIST) or REXX execs.

Besides issuing TSO/E commands, CLISTs can perform more complex programming tasks. The CLIST language includes the programming tools needed to write extensive, structured applications. CLISTs can perform any number of complex tasks, from displaying a series of full-screen panels to managing programs written in other languages.

The CLIST language is an interpretive language. TSO/E also offers a second interpretive language, REXX. REXX is a general purpose, high-level language not unlike PL/I. REXX has the usual structured programming instructions and a number of useful built-in functions.

The main difference between ICCF Procedures and Macros is that macros are executed in the foreground as normal commands and may be invoked while in edit mode, whereas procedures are executed only in command mode. Procedures require that execution of the procedure processor program be started in an interactive partition, which means that a macro is processed more quickly than a procedure. Another difference is that a procedure has more control over the flow and execution of commands than a macro.

In TSO/E, CLISTs and REXX programs are executable sequences of TSO/E commands, subcommands, and CLIST or REXX statements. The entire TSO/E command language is available to CLISTs and REXX programs.

To create a CLIST or REXX program, use the ISPF/PDF editor or the TSO/E EDIT command to put the commands, subcommands, and command procedure statements into a data set. The data set may be either sequential or partitioned. A sequential CLIST or REXX data set consists of only one program, while a partitioned data set may contain more than one program. When a PDS consists entirely of CLISTs, it is called a CLIST library. Detailed information on writing CLISTs can be found in *TSO/E CLISTs*.

7.5 Migrating from VSE/ICCF to MVS and TSO/E

As with any new system, TSO/E will require time to learn. Many of its functions are similar to those in ICCF but others are either entirely new, or differ enough that you will have to change your present methods in order to implement them. In this section we will attempt to describe how you can begin the migration from VSE/ICCF to MVS TSO/E.

7.5.1 Converting ICCF Libraries

Although there are many methods for moving members from your existing ICCF libraries to data sets accessible to TSO/E, in this section we will discuss just two. The first method is to write an ICCF procedure that will create a tape file containing the JCL and data necessary to execute the MVS utility IEBUPDTE to create a new PDS containing the members from an ICCF library. The second method utilizes the ICCF utility DTSUTIL to punch ICCF library members to tape. A program would then need to be written to reformat this tape to a format that would be acceptable to the MVS utility IEBUPDTE. The advantage of the second

method over the first would be the total flexibility available in creating the tape input to IEBUPDTE and the JCL necessary to execute this MVS utility. The advantage of the first method is its ease of implementation.

In order to write an ICCF procedure to create a ″SYSIN″ format tape for the execution of IEBUPDTE, you will need to answer questions such as the following:

- How much data will be moved on each execution of the procedure?
- How large will the new PDS have to be to hold this data?
- On what device type and volume serial will the PDS reside?
- What will the data set name be for the new PDS?
- What block size should be used on the new PDS?

The sample ICCF procedure which follows assumes that you will create a PDS corresponding to each ICCF library. It therefore unloads a single ICCF library each time it is invoked. When the procedure is executed you will be prompted for the ICCF library number you wish to unload, the TSO/E user ID, user-supplied name, and descriptive qualifier for the data set name of the new PDS, and the device type and volume serial on which the new PDS will reside. The block size for the new PDS will be 800 bytes, and 50 tracks with 10 directory blocks will be used to define the new PDS. The procedure creates an ICCF member named IEBUPDTE which is a DITTO card to tape job to be submitted to a batch partition for execution. The tape created by this job will be used on an MVS system to create a new PDS with the contents of the ICCF library.

Sample ICCF Procedure

```
************************************************************************
*
* This is an example of an ICCF procedure which could be used to
* create an MVS IEBUPDTE jobstream on tape which will create a PDS
* containing the members from an ICCF library.
*
* It creates an ICCF member named IEBUPDTE which is a DITTO card to
* tape job to be submitted to batch for execution.
*
************************************************************************
/LOAD DTSPROCS
/OPTION SAVE RESET
&&OPTIONS 00000000
&&LABEL TAG1
&&TYPE ENTER THE ICCF LIBRARY NUMBER YOU WISH TO UNLOAD
&&READ &&PARAMS
&&IF &&PARAM1 EQ ″&&GOTO -TAG1
&&SET &&VARBL1 &&PARAM1
&&SET &&VARBL2 &&USERID
&&SET &&VARBL3 ′LIB&&PARAM1′
&&SET &&VARBL4 ′DATA′
&&TYPE ENTER THE TSO/E USER ID FOR THE PDS TO BE CREATED
&&TYPE THE DEFAULT WILL BE &&VARBL2
&&READ &&PARAMS
&&IF &&PARAM1 EQ ″ &&GOTO TAG2
&&SET &&VARBL2 &&PARAM1
&&LABEL TAG2
&&TYPE ENTER THE USER-SUPPLIED NAME FOR THE PDS TO BE CREATED
&&TYPE THE DEFAULT WILL BE ICCF.&&VARBL3
&&READ &&PARAMS
```

```
&&IF &&PARAM1 EQ ” &&GOTO TAG3
&&SET &&VARBL3 &&PARAM1
&&LABEL TAG3
&&TYPE ENTER THE DESCRIPTIVE QUALIFIER FOR THE PDS TO BE CREATED
&&TYPE THE DEFAULT WILL BE &&VARBL4
&&READ &&PARAMS
&&IF &&PARAM1 EQ ” &&GOTO TAG4
&&SET &&VARBL4 &&PARAM 1
&&LABEL TAG4
&&TYPE ENTER TME DISK TYPE (IE 3350, 3375, 3380) FOR THE PDS
&&READ &&PARAMS
&&IF &&PARAM1 EQ ” &&GOTO -TAG4
&&SET &&VARBL5 &&PARAM1
&&LABEL TAG5
&&TYPE ENTER THE VOLUME SERIAL NUMBER OF THE DISK FOR THE PDS
&&READ &&PARAMS
&&IF &&PARAM1 EQ ” &&GO’TO -TAG5
&&SET &&VARBL6 &&PARAM1
&&LABEL TAG6
/SWITCH &&VARBL1
&&IF &&RETCOD EQ ’*SWITCHE’ &&GOTO TAG7
&&IF &&RETCOD EQ ’*LIB’ &&GOTO TAG7
&&TYPE USER MAY NOT SWITCH TO ICCF LIBRARY &&VARBL1
&&TYPE PROCEDURE TERMINATED
&&GOTO TAG9
&&LABEL TAG7
&&TYPE YOU HAVE REQUESTED ICCF LIBRARY &&VARBL1 TO BE UNLOADED
&&TYPE TO CREATE AN MVS JOB FOR CREATING A PDS WITH THE FOLLOWING
&&TYPE &&VARBL2.ICCF.&&VARBL3.&&VARBL4’ ON A &&VARBL5 WITH VOL
&&TYPE SERIAL &&VARBL6
&&LABEL TAG8
&&TYPE ENTER Y TO CONTINUE, C TO CANCEL, OR R TO RETRY.
&&READ &&PARAMS
&&IF &&PARAM1 EQ ’Y’ &&GOTO TAG10
&&IF &&PARAm1 EQ ’R’ &&GOTO -TAG1
&&IF &&PARAM1 EQ ’Y’ &&GOTO TAG9
&&GOTO -TAG8
&&LABEL TAG9
&&TYPE END OF PROCEDURE ICCFTSO/E
&&EXIT
&&LABEL TAGI0
&&OPTIONS 0010001
/ED
I // JOB IEBUPDTE CREATE MVS IEBUPDTE TAPE USING DITTO CT
I // UPSI 1
I * PLEASE ASSIGN SYSO20 TO A TAPE DRIVE WITH A SCRATCH TAPE MOUNTED
I // PAUSE
I // EXEC DITTO,SIZE=92K
I $$DITTO CT OUTPUT=SYSO20,BLKFACTOR=10
I //UPDATE JOB &&VARBL2
I //    EXEC PGM=IEBUPDTE,PARM=NEW
I //SYSPRINT DD SYSOUT=A
I //SYSUT2 DD DSNAME=&&VARBL2.ICCF.&&VARBL3.&&VARBL4,UNIT=&&
I // DISP=(NEW,KEEP),VOLUME=SER= &&VARBL6,SPACE=(TRK,(50,,10)),
I // DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
I //SYSIN DD DATA
TOP
STACK 13
QUIT
```

```
&&OPTIONS 1100001
/LIB FULL ALL
&&OPTIONS 0010001
/LOAD DTSPROCS
/OPTION NOPROMPT
&&OPTIONS 0010001
/LIST 1 1 IEBUPDTE
&&IF &&RETCOD NE *READY &&GOTO TAG11
/PURGE IEBUPDTE
&&LABEL TAG11
/INPUT NOPROMPT
DUMMY LINE
/END
/SAVE IEBUPDTE
/EDIT IEBUPDTE
NEXT
GET $$PUNCH
TOP
DEL 1
L SYSIN
NEXT
DEL 2
REPEAT *
O XXXXX YYYYYYYYYYYYYYYYYYYYYYYYYYYYY
LUP SYSIN
NEXT
&&NOP C'YYYYYYYYYYYYYYYYYYYYYYYYYYYYY',LEVEL=00,SOURCE=0,
LUP SYSIN
NEXT
&&NOP C'XXXXX'./  ADD NAME='*
LUP SYSIN
NEXT
ZONE 19 55
&&NOP C'" * G
ZONE 1 72
LUP SYSIN
NEXT
&&LABEL ILOOP
DUP
&&NOP C './  ADD NAME='INCLUDE'
&&NOP C ',LEVEL=00,SOURCE=0,LIST=ALL"
NEXT
&&IF &&RETCOD NE INVALID &&GOTO -ILOOP
I./ ENDUP
I $$DITTO EOJ
I /*
I /&
END
&&TYPE PLEASE SUBMIT IEBUPDTE FROM YOUR ICCF LIBRARY
```

The second method for migrating ICCF members to a PDS for TSO/E utilizes the ICCF utility DTSUTIL and an assembler program to change the output from DTSUTIL to a format acceptable to IEBUPDTE. Using the PUNCH command of the ICCF utility DTSUTIL with the PUNCTL option and SYSPCH assigned to tape, you can create a tape of selected ICCF members with imbedded ADD MEMBER and END OF MEMBER statements. These unblocked 81 byte records will become input to an assembler language program. This program will use the information from the ADD MEMBER statement to create IEBUPDTE control statements, will

delete the first character of each record, and will delete the END OF MEMBER statements. The last statement placed on the output tape will be the IEBUPDTE statement ./ ENDUP. This tape will then become the input to the IEBUPDTE utility on an MVS system. Information about the PDS to be created will be contained in the MVS JCL used to invoke IEBUPDTE.

No matter what method you choose for migrating ICCF members to data sets accessible to TSO/E, you should not attempt to move ICCF Procedures or Macros, or IPF panels and tables contained in ICCF libraries. For information on the requirements for the MVS utility IEBUPDTE, see *MVS/Extended Architecture Data Administration: Utilities*.

7.5.2 ICCF Procedures and Macros

ICCF procedures and macros cannot be used under TSO/E. It is therefore necessary to determine what function is performed by an ICCF procedure or macro and determine how this function can be implemented in TSO/E.

For example, the SUBMIT procedure of ICCF is used to submit jobs to a batch partition for execution. TSO/E provides this facility through the SUBMIT command. If you have modified the ICCF SUBMIT procedure to enforce JCL standards, you will need to investigate the exit routine capability of the TSO/E SUBMIT command processor.

Most of the function provided by IBM supplied ICCF procedures is available either through TSO/E commands, or in services provided though MVS. An example of function provided by ICCF procedures that is implemented in MVS services would be the GETL and GETP procedures. These ICCF procedures allow you to move list and punch output respectively from the VSE/POWER queues to an ICCF member. Since MVS, through the job entry subsystems (JES2 or JES3), allows you to direct your program output to a data set accessible to TSO/E, the function of these two ICCF procedures is also available to you as a TSO/E user.

If you have written your own ICCF procedures or macros to perform frequently executed sets of ICCF commands, you will find that you have the same capability under TSO/E through CLISTs or COMMAND procedures.

Chapter 8. Databases

8.1 DL/I and IMS/VS DB Differences

8.1.1 Introduction

This section addresses differences that exist between DL/I DOS/VS (Data Language/One DOS/VS) Release 1.8 and IMS/ESA (Information Management System/Enterprise Systems Architecture) Versions 5 and 6. In the context of this chapter, references to DL/I may be used interchangeably with DL/I DOS/VS or VSE DL/I.

The following matrix highlights the various functions of DL/I that will require attention during conversion.

DL/I ⟶ IMS/VS

	Areas Affected					
Functional Capability	DBD	PSB	PROG.	OPR	Util	JCL
Field-level Sensitivity	X	X	X			
Access Statement	X					
Automatic Field Start CALC	X	X				
Automatic Segment CALC	X					
RPG II			X			
Command Level (HLPI)			X			
Secondary Indexes				X	X	X
Selective Unload				X	X	X
Disk Logging				X	X	X
UPSI			X	X		X
Buffer Specification				X		X
Parameters			X	X		X

Figure 15. DL/I Functions Requiring Attention when Migrating to IMS/VS

8.1.2 MVS System Requirements

IMS/ESA requires a Type 2 SVC in the MVS nucleus and a Type 4 SVC in LPALIB.

IMS/ESA Resource Clean-up Module and ABEND Dump Formatting Routine must be link-edited into SYS1.LPALIB.

IMSESA.RESLIB must be APF authorized.

8.1.3 Data Base Descriptor (DBD)

1 Automatic field start calculation

Must code START= in the IMS/ESA DBD.

2 Automatic segment length calculation

Must code BYTES= in the IMS/ESA DBD.

3 FBA DASD Support

MVS does not support FBA (FIXed Block Architecture) DASD devices. This requires changing the DEVICE= parameter of the DATASET statement to the device type that will be used.

4 ACCESS Statement

The ACCESS statement is not supported in IMS/ESA. All parameters of the ACCESS statement have equivalent function through parameters of the DBD. This is provided in either DL/I or IMS/ESA DBDs. Therefore, the DBD should be used for portability between DL/I IMS/ESA. The three types of ACCESS statements and the required changes are:

- ***HDAM DB***

Access Stmt.	DBD equivalent (DL/I or IMS/ESA)
RMRTM=	RMNAME= (module,
CIANPT	# of root anchor points,
PRIMCI=	# of CIs in root addressable area,
RILIM	record insert limit in bytes)
SEGM=	data base root implied
SEQFLD=	SEQFIELD of root implied if PARENT=0 is coded
SEQVAL=	third subfield of the NAME parameter in the FIELD statement defining the sequence field

- ***Primary Index of HIDAM DB***

Access Stmt.	DBD equivalent (DL/I or IMS/ESA)
REF=	separate DBD with ACCESS=INDEX needed
SEGM=	NAME= of LCHILD in index DBD
SEQFLD=	INDEX= of LCHILD in index DBD

In addition an LCHILD must be coded in the data DBD referencing an index DBD.

- ***Secondary Index for HD DB***

Access Stmt.	DBD equivalent (DL/I or IMS/ESA)
REF=	separate DBD with ACCESS=INDEX needed
SEGM=target seg	determined by which SEGM the LCHILD and XDFLD follow in data DBD
SEQSEG=source seg	SEGMENT= of XDFLD

SEQFLD=	SRCH= of XDFLD
SEQVAL=	SUBSEQ= of XDFLD
SUPVAL=	NULLVAL= of XDFLD
SUPRTN=	EXTRTN= of XDFLD

The non-ACCESS statement approach consists of an LCHILD and XDFLD following the target segment, SEGM statement, in the data portion of the database. Associated FIELD statements are needed in data DBD if /sx or /ck are used. An index DBD defining the index database with its LCHILD statement referencing the target segment in the database is also required.

8.1.4 Program Specification Block (PSB)

The following will require changes to the PSB.

- The PSB LANG= parameter must be COBOL, PL/I, PASCAL, ASSEM, or blank for IMS/ESA

  No trailing blanks are permitted. Alternative spellings of ′PL.I′ will have to be changed.

  Supported languages include ADA, C, OS/VS/COBOL, VS COBOL, C/370, Pascal, PL/I, RPG/370, REXX, and Assembler.

- Automatic Field Start Calculation SENFLD

  IMS/ESA requires coding of START= parameter in SENFLD of PCB.

- The following field level sensitivity extensions of DL/I are not available in IMS/ESA. If you have used these functions, program changes may be required.

  1. Virtual Fields
  2. Field Types Z,E,D,L
  3. Automatic Data Conversion
  4. Field Exit Routines

8.1.5 Batch Programming

8.1.5.1 RPG II

These applications will need to be converted to RPG/370 or some other IMS/ESA supported language.

8.1.5.2 Interactive Macro Facility (IMF)

DL/I DOS/VS Interactive Resource Definition and Utilities (5746-XX1) provides the Interactive Macro Facility (IMF) and Interactive Utility Generation (IUG) functions. MVS ISPF/PDF may be used for these functions. The specific panels and worksheets provided by IMF and IUG are not provided by MVS ISPF/PDF.

8.1.5.3 Command-Level Coding (HLPI)

Assembler, PL/I and COBOL are supported. CICS supplies the translator.

8.1.5.4 Statement Compatibility

All batch programs using the calls, GU - GHU - GN - GHN - GNP - GHNP - ISRT - DLET - REPL, are transportable to MVS with no modification to the DLI calls. Of course, these programs may need changes for other reasons and they must be recompiled on the MVS system. VSE JCL is changed to MVS JCL, and the DL/I parameters external to the program (that is, HSBFR and HDBFR bufferpool) are written differently even though they perform the same functions. They are defined in the IMS/ESA control data set DFSVSAMP. This is described in the *IMS/ESA Installation Volume 2: System Definition and Tailoring* manual.

The languages common to DL/I and IMS/ESA are PL/I, COBOL/VS and Assembler. RPG II is not supported by IMS/ESA, but RPG/370 is.

The status codes tested with DL/I are valid with IMS/ESA and have the same meanings with some exceptions which are covered below.

8.1.5.5 Field Level Sensitivity

Basic support is compatible between DL/I and IMS/ESA. If DL/I extensions are used, changes may be required to application programs as well as to database definitions. Status codes starting with ′K ′ will not be returned by IMS/ESA.

8.1.5.6 PCB after GE Status

There is a difference in the information returned with the GE status code following a number of GNP calls. In DL/I the segment name returned is that of the last successfully retrieved segment. In IMS/ESA, the segment name returned with the GE status code is that of the parent segment.

8.1.5.7 NI Status Codes

DL/I requires a user to manually correct a database following an NI status code. IMS/ESA automatically fixes this error condition.

8.1.5.8 CHKP Calls

While the written form of the CHKP call is the same, the PCB-name specified in IMS/ESA must refer to the first PCB in the PCBLIST. This PCB is called the I/O PCB and is generated with the CMPAT option in PSBGEN. Since this is a new PCB, application programs will have to be changed to reflect the existence of this PCB.

8.1.5.9 GSCD and/or GSTA Calls

These must be reviewed for the following reasons:

1. GSCD has a common format and returns the address of the SCD area. However, the SCD DSECT layout is not the same and label names may differ.
2. DL/I supports the GSTA call for application programs running across separate VSE/VAE address spaces. This allows access to DL/I statistics when DL/I Multiple Partition Support (MPS) applications are running in separate VAE address spaces. The IMS/ESA STAT call provides similar functions.

8.1.5.10 Assembler Language Calls

CALLDLI MF=E is not supported in IMS/ESA.

8.1.6 Utilities

Equivalents of all DL/I utility programs exist in IMS/ESA. Their functions are similar, but it is necessary to change the JCL from VSE to MVS and utility control statements from DL/I to IMS/ESA. There are variations in the utilities between DL/I and IMS/ESA that will require procedural changes.

- Partial HD Reload is not supported in IMS/ESA.
- Selective HD Unload is not supported in IMS/ESA.

Similar function is provided by the IMS System Utilities/Data Base Tools (5685-093).

8.1.6.1 Rewind Option for Reorganization Utilities

This is controlled by JCL in MVS. The VOLUME parameter on the tape DD statement allows you to specify whether or not the tape is to be dismounted. The LABEL parameter indicates how many data sets precede the required data set on the tape volume.

8.1.6.2 Checkpoint Option with HD Reorganization Unload Utility

In IMS/ESA, checkpointing is requested through the DFSUCKPT DD JCL statement. Restart requires the DFSURSRT DD statement as well. See the *IMS/ESA Utilities Reference: Database Manager* manual for more information.

8.1.6.3 Secondary Index Creation

In DL/I the secondary indexes are created when the data portion of the database is loaded. DL/I will create a secondary index during load, HD reload, or prefix update. IMS/ESA uses the HISAM unload and reload utilities to create the secondary indexes from a work data set created by the prefix resolution utility. The Prefix Resolution and HISAM unload and reload utilities are run after the data portions of the database have been created. Refer to the *IMS/ESA Administration Guide: Database Manager* and the *IMS/ESA Utilities Reference: Database Manager* manuals for the reorganization of databases with secondary indexes or logical relationships.

8.1.7 Operations

8.1.7.1 RESTART with CHKP

If you are using CHKP in your DL/I batch jobs and have developed procedures for restarting that are acceptable to MVS, do not make any unnecessary changes at this time. If your existing procedures require more than minimal conversion effort, you should consider the use of the IMS/ESA symbolic checkpoint and IMS GSAM. All new IMS/ESA applications should use the symbolic form instead of the basic form of CHKP. One exception would be applications using CICS/OS Shared Database.

8.1.7.2 Backout Utility/Disk Logging

IMS/ESA supports both DASD and tape logging in batch. The archive utility DFSUARC0, is used to copy disk logs to tape. Disk is acceptable as input to all IMS/ESA utilities including backout.

8.1.7.3 UPSI

The use of the UPSI byte is not supported by MVS. The equivalents for the various bits are:

- Bit 0: Reading of parameter statement.

  This function is replaced with parameters in the EXEC statement of the MVS JCL.

- Bits 1-3: Available to the application programmer.

  If these bits have been used, the application programs must be modified to the MVS standard established for the installation.

- Bit 5: Storage dump request if STXIT active.

  The presence of the SYSUDUMP DD statement determines if a dump is to be taken in MVS.

- Bit 6: Log function active or inactive.

  The IEFRDER DD statement being set to DD DUMMY is the only way to avoid logging with IMS/ESA. If Data Base Recovery Control (DBRC) is used, logging is forced for all batch problems using a PSB with update intent on any DBRC registered database. Thus, you may not specify IEFRDER DD DUMMY.

- Bit 7: Setting STXIT active or inactive.

  There is no direct equivalent required in MVS. The SPIE= parameter in the IMS/ESA EXEC statement relates to the handling of an application program SPIE during a CALL. For more information see the *IMS/ESA System Programmers Guide*, procedures DBBATCH and DLIBATCH.

8.1.7.4 DL/I Parameter Statement

The functional equivalents of this statement in IMS/ESA are implemented through either EXEC statement PARMs or entries in the DFSVSMnn member of IMSESA.PROCLIB. This is located through the DFSVSAMP DD statement. The following applies only to VSAM databases. For OSAM options and more detail on VSAM, see the *IMS/ESA Installation Guide* and the *IMS/ESA System Administration Guide*.

DL/I Parameter	IMS/ESA Equivalent
progname	PARM on EXEC
psbname	PARM on EXEC
buf	# entries DFSVSMnn
HDBFR	VSAM Subpool statement in DFSVSMnn
HSBFR	Uses VSAM Subpool
TRACE =	DFSVSMnn OPTIONS or/and DLITRACE
ASLOG =	OPTIONS LTWA=
LOG =	IEFRDR DD data set
RC=	not supported

8.1.8 Database Portability

There are two fundamental approaches to making your DL/I databases available to IMS/ESA. One is to unload the database using DL/I utilities and reload it using IMS/ESA utilities. The other is to ″position″ your DL/I databases using DL/I and IMS/ESA compatibility options in the DBD during a normal database reorganization under DL/I.

Since database reorganization involves considerable processing for large and complex databases, ″positioning″ may offer some advantages. It can reduce the time required to actually switch the production applications from VSE to MVS. It can also allow alternate, not concurrent, access to the database by DL/I and IMS/ESA applications. Both approaches are described below.

8.1.8.1 Alternate DL/I and IMS/ESA Access

A database may be alternately accessed by VSE and MVS if the physical organization meets certain restrictions and compatibility options are specified in the appropriate DBDs. This allows IMS/ESA to access and update a DL/I database. However, the database must be initially unloaded and reloaded using DL/I utilities. If any reorganization is desired, this must also be performed under DL/I. IMS/ESA utilities may be used to make IMS image copies, merge IMS log tapes and execute IMS backout or forward recovery operations. Note that the image copy tapes, change accumulation tapes, or logs are not portable between DL/I and IMS/ESA.

To establish this environment, compatibility options must be used.

1. Unload the database using the VSE DL/I utility.
2. Identify and resolve any VSE VSAM incompatibilities as described in 5.6, “VSAM Differences” on page 110.
3. Specify **IMSCOMP=YES** in the DL/I DBDs describing the data portions of the database. Note that changes to the VSAM logical record sizes will be required. These are identified during the DBD generation.
4. Regenerate the DBDs and ACBs in VSE.
5. Reload the database using the appropriate VSE DL/I utility. The database may now be accessed by VSE applications again.
6. Specify DOSCOMP in the IMS/ESA DBDs describing primary or secondary index portions of the database.
7. Generate the IMS/ESA DBDs under MVS. The database is now accessible by IMS/ESA whenever the VSAM user catalog defining it has been disconnected from VSE and connected to MVS.

The database must be on a compatible VSAM supported device such as 3380, 3350 or 3375. FBA devices such as 3310 and 3370 are not supported by MVS.

When running in compatibility mode make sure the IMS/ESA and DL/I DBD definitions are kept in synchronization. Do this by explicitly defining parameters rather than letting them default. For example, CI size and record length for IMS/ESA should be defined to be the same as the DL/I values.

Once migration is complete the DOSCOMP parameter may be removed when the database is reorganized using the IMS/ESA utilities. There is no need to unload under DL/I and reload under IMS/ESA in this case.

Recovery of DL/I - IMS/ESA compatible databases require special procedures. Image copy, change accumulation, and log files are not compatible between DL/I and IMS/ESA. The recovering of database changes performed under DL/I must be performed using DL/I image copy, change accumulation, and log files with DL/I utilities; and the recovering of database changes performed under IMS/ESA must be performed using IMS/ESA image copy, change accumulation, and log files with IMS/ESA utilities. As such, one would probably want to take an IMS image copy prior to switching to IMS/ESA update access of the databases and a DL/I image copy prior to switching to DL/I update access of the databases.

8.1.8.2 Unloading and Reloading the Database

Should the physical database need to be migrated to MVS, and there is no requirement to access the database with DL/I after it has been converted to IMS/ESA format, the following steps should be followed:

1. Perform an unload using DL/I.
2. Translate and compile the DBDs, PSBs, and ACBs using IMS/ESA.
3. Perform a reload using IMS/ESA.

As there is no HDR2 label record on the standard tape label created by the DL/I unload utility, the DD statement for tape input to the IMS/ESA reload utility must specify blocksize and recordsize. See the *DL/I Data Base Administration Guide* for recommended values.

Note that secondary index creation is different for IMS/ESA (see 8.1.6, “Utilities” on page 173).

The HD unload/reload must be used for HDAM, HIDAM, and HISAM.

Tape files must be used across systems and not disk when unloading and reloading. The DL/I HISAM unload cannot be used as input to the IMS/ESA HISAM reload.

You should not attempt to unload from IMS/ESA and reload using DL/I.

The AMS recordsize parameter for IMS/ESA should be CI size-7 rather than CI size-10 when you define the cluster to VSAM. The IMS/ESA DBD generation will provide suggested parameters. The segment prefix for index databases is one byte shorter for IMS/ESA.

The following flow chart depicts the steps to be taken in migrating your databases, with or without compatibility, and in generating your IMS/ESA system.

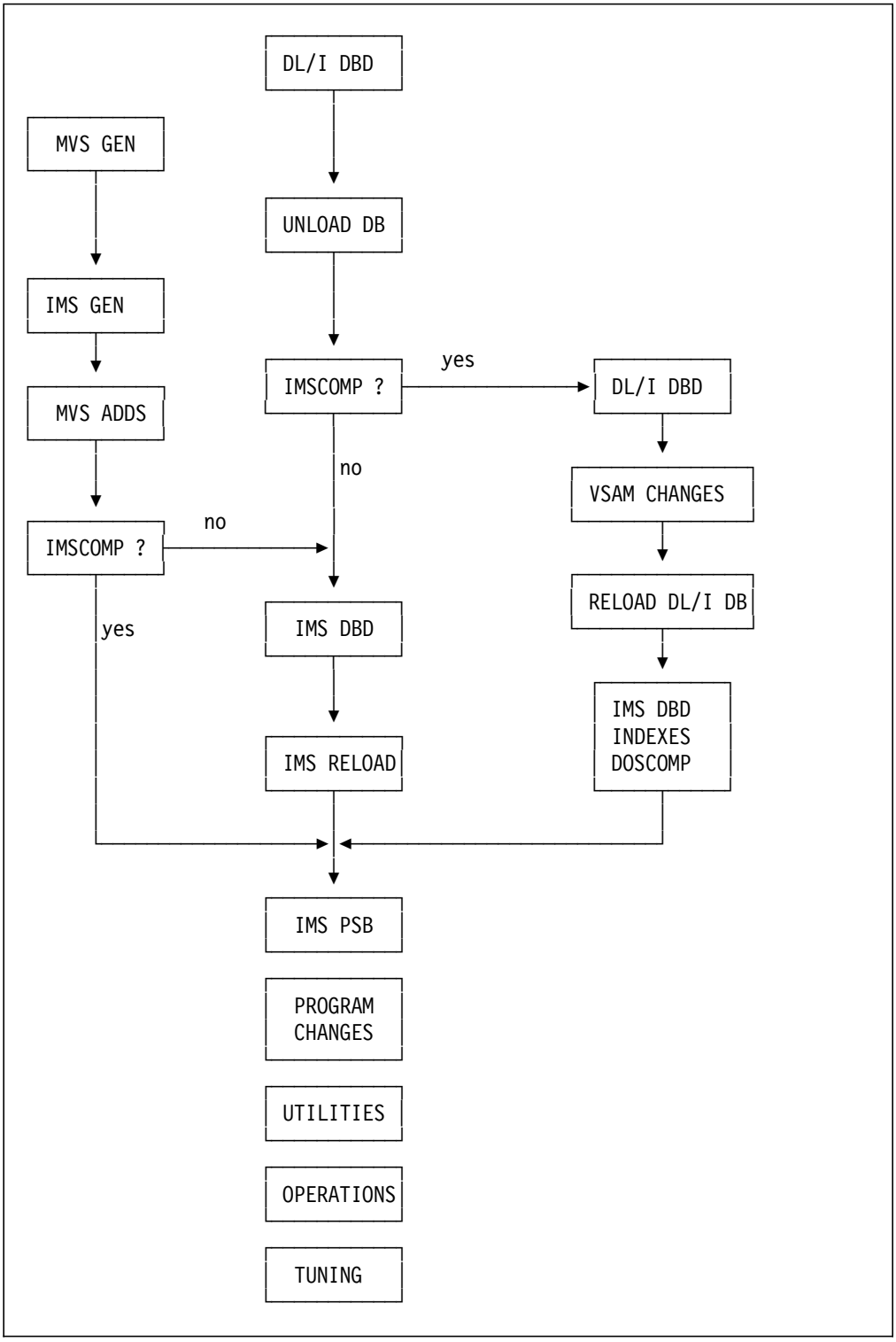


Figure 16. Steps in Migrating DL/I Databases to IMS/ESA

8.1.9 DL/I Multiple Partition Support

Conversion to IMS/ESA BMPs (Batch Message Processing programs) running under DBCTL should be considered as an alternative to CICS/OS Shared Data Base or IMS/ESA Data Sharing support.

8.1.10 Additional Information

A recently announced Redbook, *Interoperability between VSE DL/I and OS/390 IMS DBCTL*, SG24-5249, can provide additional conversion information.

8.2 SQL/DS to DB2 for OS/390 Migration Consideration

Note: Although the formal name of the SQL/DS product has changed to DB2 for VSE, this document will use the name SQL/DS. This document will also use the term ′DB2′ to mean the full product name - ′DB2 for OS/390′.

8.2.1 Descriptions of Users

The differences and thus the migration considerations between SQL/DS and DB2 take on meaning only as they pertain to, or are perceived by, the users of the products. In order to discuss this, we need to define who these users are. The type of users we want to address are:

- End Users
- Application Developers
- Data Base Administrators (DBAs)
- System Administrators
- Security Administrators

It is important to point out at this time that the following is not meant to provide an exhaustive list of differences between the two products. Instead, it is intended to point out the most likely areas of difference you will encounter to give a feeling for how significant or insignificant these differences may be. An exhaustive treatment and explanation of the differences in the VSE and OS/390 platforms is given in the *IBM SQL Reference*, SC26-8416. This can be ordered through standard IBM document ordering procedures.

As we will see later, the area of most concern is with the language of both products - the Structured Query Language or SQL. SQL has the three following ′flavors′:

1. Data Manipulation Language (DML) that is used by End Users and Applications Developers
2. Data Definition Language (DDL) that is used by DBAs and Systems Administrators
3. Data Control Language (DCL) that is used by Security Administrators.

8.2.1.1 End Users

From the standpoint of end users, for DML, there are very few differences between SQL in the two products. DML is what most end users use - the issuance of SELECT, UPDATE, INSERT and DELETE statements to do work against a database. End users rarely have a need to know or use DDL or DCL. As we will see later, the latter two SQL ′flavors′ are used mostly by DBAs,

systems and security folks. This is not to say that differences do not exist, just that they are minor and should not be perceived by most end users. QMF users should see virtually no differences. The one possible exception is the TRANSLATE scalar function which is not supported by DB2.

8.2.1.2 Application Developers

While end users will see little or no differences, application developers can expect to see more differences between the two products. However, it is important to realize that the basic job of developing an application is the same.

- Developers can use the same design process.
- The method of making an SQL call in a program is the same (through the use of the EXEC SQL statement).
- The operators - DECLARE, OPEN, FETCH and CLOSE - are the same. These are statements used in application programs to handle cursors.
- Application program preparation using precompile and bind is essentially the same - very few differences.

Both products have facilities to enter one or more SQL statements at a terminal for execution against a relational database. This could be done for testing or to do some work without having to write a program, for example, a one time requirement. In SQL/DS this facility is called *'ISQL'* that runs under CICS/VSE and in DB2 it is called *'SPUFI'* and runs under ISPF in TSO. SPUFI does not have the limited answer set formatting and printing capability of ISQL.

Application developers should consider that while the program preparation process is similar, some differences exist. For example, DB2 preparation occurs in two phases - precompile and bind. For SQL/DS both phases are done in the same step. Because of DB2′s separation of these functions, the database does not need to be available during the precompile phase. SQL/DS does have the ability via the DBSU to UNLOAD and LOAD packages. DB2 does not have a utility capable of unloading and loading a package. If you are using this SQL/DS capability, you will need to save the DB2 DBRM or preprocess the program again to create the DB2 DBRM.

For application programming languages, DB2 supports all of the languages supported by SQL/DS except RPG. It should be noted however, that SQL/DS V3R5 was the last release which provided this support; it was dropped in V5.

There are some minor variations in the application programming interfaces for the two products. For example, the meaning of SQLCODE values other than 100, 0, -803, and -911 is product specific. Thus, how an application program′s logic handles SQLCODEs other than these will need to be examined and modified appropriately. A major improvement was provided with the introduction of DB2/VSE V5: SQLSTATEs are now at the SQL92 level as is DB2/OS390. That means that an application can use the SQLSTATE to determine an error and that the SQLSTATE is **no longer** platform specific.

Besides differences in SQL return codes, other differences exist in the SQLCA (SQL Communications Area) following a negative SQL return code. The SQLCA has a number of fields that indicate the condition of the most recently executed SQL statement. You should update your program logic that processes, records, and displays this information.

8.2.1.3 Database Administrators (DBAs)

As with the previous two groups discussed, there are really more similarities than differences between the two products for the DBA. The design methodology is the same, whether you prefer normalization or some other technique. The optimizer selects access paths basically the same way, relieving the DBA from making this choice. Differences show up in some of the detail work of database design. Differences exist in the areas of database object sizes, SQL limits, locking levels, DRDA considerations, referential integrity definitions, and data formats. SQL/DS is a subset of DB2 in these areas except an SQL/DS table definition can contain one or more LONG VARCHAR columns that could materialize an actual row length that exceeds the 32K limit of a DB2 row. If you do have this situation, you will need to redesign this table to fit within the 32K row limit of DB2 and modify the programs appropriately that use this data.

8.2.1.4 System Administrators

There are more differences in this area than in the previous areas. This is primarily due to environmental differences in the VSE and the OS/390 platforms. Both systems provide the entire complement of systems management, utilities and integrity services expected of a robust RDBMS. However, implementation may be different.

In both systems, normally all database changes caused by the execution of SQL statements are logged. However, when running SQL/DS in a single user environment this logging is optional and when you have an SQL/DS storage pool with the NOLOG attribute this logging is not done. Both of these capabilities are not available in DB2 and will have to be evaluated and mapped to a different DB2 implementation.

Since recovery capabilities and procedures are very different between SQL/DS and DB2, you must re-evaluate your needs and design a recovery procedure using the DB2 facilities. This also applies to Utilities.

The following is provided to help you in mapping SQL/DS utility functions to DB2 utility functions:

- Reorganize a table
  - SQL/DS - DBSU UNLOAD/RELOAD
  - DB2 - REORG utility
- Update table statistics
  - SQL/DS - UPDATE STATISTICS SQL statement
  - DB2 - RUNSTAT utility
- Load data into a table
  - SQL/DS - DBSU LOAD
  - DB2 - LOAD utility
- Unload data to a user defined sequential file
  - SQL/DS - DBSU DATAUNLOAD
  - DB2 - not supported

While both SQL/DS and DB2 operate from the same relational basis of viewing data as tables, the underlying storage structures used differ. At the logical level, a DB2 database is composed of one or more tablespaces which in turn are composed of one or more tables, and indexspaces, which contain one index. In SQL/DS, both tables and indexes are stored together in dbspaces.

At the physical level, in DB2 each tablespace is stored in a pageset, which consists of one or multiple VSAM ESDS data sets or LDSs. In SQL/DS, data is stored in storage pools that are composed of one or more dbextents. In VSE, a dbextent consists of one VSAM ESDS data set. While this may seem similar, the difference is that in DB2 one VSAM data set is used for only one tablespace. In SQL/DS, one VSAM data set can be used to store data from any or all tables in the system (within one storage group).

As a result of the different storage structures and names used, a new physical data model must be developed for the DB2 environment and then the parameters associated with storage structures in the data definition language (DDL) SQL statements must be updated to implement the new physical data model.

8.2.1.5 Security Administrators

Both products have the same set of table privileges (ALTER, DELETE, INDEX, INSERT, REFERENCES, SELECT and UPDATE). Other privileges related to the use of resources (such as using space by creating table) and operation (such as executing programs) are granted in a similar manner, but the definition of user types is different. Thus, there are different operands within the GRANT command. DB2 allows group authorization (not in SQL/DS, but in the new *Control Center* feature of DB2/VSE V5) so that fewer GRANT statements need to be issued.

Implementation is different in how users are given certain authority and each product has its own naming convention. Because the two products differ in how databases are defined and organized, the levels of authorization are more robust in DB2. In DB2, the highest level authority is called SYSADM and in SQL/DS, the equivalent level is called DBA. Some noteworthy differences in the SQLs are:

- The VSE CONNECT statement can have ″user IDENTIFIED BY password″, but the OS390 can not.
- Tables on VSE can specify CCSIDs at the column level. Current (for DB2/OS390 V5) CCSIDs can only be specified at the table level. So if the VSE customer is utilizing this, they will need to convert their data to use a common CCSID in the table.

When DB2 is installed, only users with SYSADM authority have the SELECT privilege on catalog tables. With SQL/DS, upon initial install all users have SELECT privilege on catalog tables. In DB2, SYSADM and other users granted the proper authority can update catalog statistics used by the optimizer in access path selection. In SQL/DS, DBA authority includes all table privileges on catalog tables. The tables in the catalog are organized differently in the two products. Therefore, catalog queries will be incompatible between the two.

8.2.2 Other Comparison Areas

8.2.2.1 Year 2000

DB2 for OS/390 Version 5 is Year 2000 compliant. For DB2 for MVS/ESA Versions 3 and 4, you need to apply an APAR for Year 2000 compliance. For more details, get the DB2 and the Year 2000 White Paper from the Web. Its URL is:

- http://www.software.ibm.com/year2000/db2-html

SQL/DS Version 5 (proper name is DB2 for VSE Version 5) is Year 2000 compliant.

8.2.2.2 DRDA Considerations

DRDA level of functions in SQL/DS are upward compatible to DRDA functions in DB2. DB2 has more DRDA functions than SQL/DS. For the DRDA Remote Unit of Work (RUW) level, DB2 is compliant as an Application Requestor (AR) and Application Server (AS). SQL/DS is only on the AS level for RUW. For the DRDA Distributed Unit of Work (DUW), DB2 is AR and AS compliant, again SQL/DS is only at the AS level. Stored Procedures in DB2 can be used to do DRDA AR and AS level work, SQL/DS has no such support.

8.2.2.3 Data Replication and Data Access

DB2 has an Apply and Capture component for DataPropagation - both are separate products. SQL/DS only has a Capture component as a feature of the SQL/DS product. Both DBMSs can be accessed by DataJoiner. DB2 can interface with DataPropagator NonRelational, SQL/DS has no interface to this product. Both DBMSs have interfaces to the DataRefresher product to do data extractions.

8.2.2.4 Transaction Management

SQL/DS can interface to CICS/VSE and ICCF. DB2 can have the front-ends of IMS/ESA, CICS/ESA or TSO.

8.2.2.5 Other Product Areas

Both products can be accessed from the internet using Net.Data. SQL/DS has a VSAM Transparency product, DB2 does not. Both can be used with DataHub. In the area of administration tools, SQL/DS has a tool called SQL Master. For DB2, there is the Automated Utility Generator, and DB2 Administrative Tool. QMF can be used to query both DBMSs.

8.2.3 Summary of Migration Task

It is difficult to be precise as to what are the specific tasks and their order for a migration effort, but we can give you general ideas.

1. Acquire DB2 for OS/390 skills
2. Install DB2 and prerequisites products (at supported levels). Then install other companion products of your choice (in the area of transaction management, data replication, administration and so on). Next you need to tailor your DB2 system by specifying DB2 parameters and execution options.
3. Change SQL syntax where required. This will probably not be a big effort for DML (programs and user queries). For DDL and DCL, this is a different story and there is a bigger difference in these two areas of security administration and data and DB2 object definition. The SQL Reference document will be very helpful here.
4. Migrate object definitions to DB2
5. Move the data by unloading the data from SQL/DS and then doing a load into DB2
6. Migrate the application programs

7. Migrate the user queries
8. Migrate user profiles and authorizations
9. Change operational procedures to reflect new backup/recovery, problem determination and startup/shutdown procedures

Chapter 9. Telecommunications Subsystems

VSE and OS/390 platforms rely on the same set of communications products and protocols. Although the product sets are the same, some differences exist in product implementation and usage. The differences are primarily related to the operating systems and their file structures. This chapter will discuss the similarities and differences between VSE and OS/390 environments for the following telecommunications products:

- 9.1, ACF/VTAM
- 9.2, ACF/NCP
- 9.3, BTAM
- 9.4, Migrating TCP/IP
- 9.5, MQSeries

9.1 ACF/VTAM

The most recent release of VTAM for OS/390 is VTAM Version 4 Release 4.1. However, this product is not available separately and is packaged together with TCP/IP as the OS/390 eNetwork Communications Server Release 5. Previous releases of VTAM up to Version 4 Release 4 were available as a stand-alone product, 5695-117. At the time of writing, supported releases of VTAM for VSE include Version 3 Release 4 and Version 4 Release 2.

The functions of VTAM for OS/390 are, with one major exception, a superset of the functions of VTAM for VSE. The main differences are related to those between the operating systems, rather than to any differences in the way that SNA networks are defined and configured. However, OS/390 does not provide a facility such as the VSE Interactive Interface (II) to guide you through SNA resource definitions. You must code new network resource definitions from scratch, although you should be able to use most of your existing VSE definitions without change. The exception would be the definition of new resources required for networking configuration changes associated with the migration project.

The following subsections summarize the tasks you will have to perform to install and set up an OS/390 VTAM system. For further details we recommend the following product manuals (those quoted are for Communications Server for OS/390 Release 5):

- *eNetwork Communications Server for OS/390 Installation and Migration Guide*, SC31-8622
- *eNetwork Communications Server for OS/390 Network Implementation Guide*, SC31-8563
- *eNetwork Communications Server for OS/390 Resource Definition Reference*, SC31-8565
- *eNetwork Communications Server for OS/390 Resource Definition Samples*, SC31-8566
- *eNetwork Communications Server for OS/390 Programming*, SC31-8573
- *eNetwork Communications Server for OS/390 Customization*, LY43-0110

9.1.1 Product Installation

The VTAM installation procedures for OS/390 are very different from those for VSE, since this is the area where the operating system differences are most influential. To install OS/390 VTAM you must perform the following steps:

- Allocate data sets for VTAM. OS/390 VTAM can make use of a large number of data sets, depending on the options selected. In 9.1.1.1, “VTAM Data Sets” we describe the essential ones; for a complete description (including the optional ones used for APPN, CMIP management and so on) please refer to the *VTAM Installation and Migration Guide*.
- Define the channel-attached VTAM devices (37XX, 3174, 3172, channel-to-channel connections and so on) in the OS/390 generation procedure. Alternatively, OS/390 allows these devices to be defined dynamically to its I/O configuration but you must make sure that this is done every time OS/390 is restarted.
- Determine the CSA and ECSA storage to be used by VTAM, and make appropriate changes to the OS/390 startup definitions. To work out how much storage your VTAM address space will use, go to the Web site at `www.ibm.networking.com/vtaprod/vtastor.html` where you will find an interactive application that does it for you.
- Install VTAM from the product tapes using SMP. If you have received VTAM as part of OS/390 or Communications Server/390 this may already have been done. Note that some VTAM modules are linked into the OS/390 nucleus, so a new VTAM installation requires an OS/390 restart. Subsequent updates and fixes may not require a system restart, since most of the VTAM code is now loaded from link and LPA libraries.
- Create a VTAM start procedure in SYS1.PROCLIB (or another procedure library known to JES). Figure 17 on page 187 shows a sample *working* procedure that contains all the essential data sets used by VTAM.
- Copy and modify your VTAM definitions and tables into the new OS/390 libraries.

9.1.1.1 VTAM Data Sets

The data sets that VTAM must have available in order to run are summarized below. They may have any *data set* names you wish to assign to them, but VTAM recognizes most of them by their *data definition* (DD) names so that is how we identify them. Figure 17 on page 187 shows a VTAM start procedure that refers to them.

1. The VTAM load modules are loaded from SYS1.LINKLIB (or an authorized library concatenated to SYS1.LINKLIB), and from SYS1.LPALIB. These data sets are not specific to VTAM, and do not need to be referenced in the VTAM start procedure. The VTAM initialization module is read from SYS1.LINKLIB and the rest of the VTAM product modules are read into the link pack area from SYS1.LPALIB.
2. SISTCLIB contains the VTAM modules which are loaded into CSA and ECSA. This data set is a PDS containing load modules, and must be allocated with DCB=(RECFM=U,BLKSIZE= whatever optimum block size your installation has decided upon). In our procedure we have used the default name of SYS1.SISTCLIB.
3. STEPLIB contains those modules which are not part of VTAM, but which VTAM uses to manage the network. Examples might include the NCP loader

(from ACF/SSP) and user-written VTAM exits. In our example STEPLIB points to the ACF/SSP library which contains the NCP loader. Any libraries defined by STEPLIB are PDSs containing load modules, and have similar DCB characteristics to SYS1.LINKLIB.

4 VTAMLIB contains VTAM tables which are assembled and linked, such as the subarea Class of Service table and the various mode tables. Again, VTAMLIB is a load module PDS with DCB=(RECFM=U) and an appropriate block size. In our example four libraries are concatenated together; the IBM-supplied VTAM tables are in SYS1.VTAMLIB and any user-defined replacements are in the two other libraries defined in front of SYS1.VTAMLIB.

5 VTAMLST contains VTAM tables which are *not* assembled and linked. Here reside the definitions of the VTAM resources (major nodes), the NCP source deck, VTAM start options and those tables which VTAM reads in source form. VTAMLST is a PDS with DCB=(RECFM=FB,LRECL=80) and a suitable block size. In the example three VTAMLST data sets are concatenated together; all the VTAM definitions are in these libraries and SYS1.VTAMLST is not used.

6 The NCPLOAD DD statement points to the data set(s) where NCP load modules may be found. The DD name for this statement is user-defined, and VTAM discovers it from the NCP source definitions (BUILD LOADLIB=).

7 In addition, VTAM requires a member IVTPRM00 to be present in the SYS1.PARMLIB OS/390 data set. This member provides initialization parameters for Communication Storage Manager (CSM), which is part of VTAM from V4R4 onwards. CSM provides data storage facilities for both VTAM and TCP/IP, as part of the high-performance data transfer function implemented by those products on high-speed connections.

```
//NET    PROC  PERF=13
//NET    EXEC  PGM=ISTINM01,REGION=6000K,TIME=1440,DPRTY=(15,13),      C
//       PERFORM=&PERF
//STEPLIB DD DSN=SSP.V4R6.SSPLIB,DISP=SHR
//VTAMLIB DD DSN=SA39.VTAMLIB,DISP=SHR
//         DD DSN=ITSC.VTAMLIB,DISP=SHR
//         DD DSN=SYS1.VTAMLIB,DISP=SHR
//         DD DSN=SYS1.NETVIEW.V3R1M0.SCNMLNK1,DISP=SHR
//VTAMLST DD DSN=ITSC.VTAMLST,DISP=SHR
//         DD DSN=BUCZAK.VTAMLST,DISP=SHR
//         DD DSN=RISC.VTAMLST,DISP=SHR
//SISTCLIB DD DSN=SYS1.SISTCLIB,DISP=SHR
//NCPLOAD DD DSN=ITSC.NCPLOAD,DISP=SHR
//         DD DSN=RISC.NCPLOAD,DISP=SHR
```

Figure 17. VTAM Start Procedure

9.1.2 Resource Definition and Operation

The differences in the coding of VTAM definitions, tables and start options are minor, but cannot be ignored. The *VTAM Resource Definition Reference* and the *VTAM Network Implementation Guide* should be used to review operating system differences.

The current supported levels of VTAM on VSE are V3R4 and V4R2. VSE VTAM V4R2 is available in three different functional level packages:

- Client/Server
- MultiDomain
- InterEnterprise

Packages are priced according to the amount of function provided and are password enabled via the VTAM start procedure.

The eNetwork Communications Server for OS/390 is only available in a single flavor, which is the functional equivalent of the high end VSE/VTAM package (InterEnterprise). The four main differences between VSE/VTAM and OS/390 VTAM are as follows:

1 VSE/VTAM supports Integrated Communications Adapters (ICAs) on 9221, 937X and 43XX processors.

These adapters are not supported by OS/390 VTAM, and alternate methods of connecting your network to a channel must be found. The various ICAs and possible alternatives follow:

- Multi-Protocol Communication Subsystem (6251 - 6254)
  - ► IBM eNetwork Communications Server for OS/2 (prod #4301114) or Windows NT (prod #4231747) with a Wide Area Connector (WAC) card or Multi-Protocol Adapter (MPA) card can provide support for SDLC communication lines.
  - ► The 3172 Interconnect Controller, in addition to LAN connectivity can provide support for up to 32 SDLC communication lines.
  - ► The 2210 and/or 2216 multi-protocol routers can be used to support multiple line protocols including SDLC. These routers can be attached to the host via a LAN gateway (such as the OSA-2 adapter). In addition, the 2216 can be channel attached much like the 3172.
  - ► A 37XX controller with NCP can provide support for SDLC line protocols for installations required to support larger numbers of remote devices.
- Asynchronous Communication Subsystem (6241 - 6244)
  - ► A 37XX controller with NCP/PEP can provide connectivity for asynchronous Start/Stop (TTC2) devices.
- ASCII Subsystem (6245 - 6248)
  - ► Although MVS/VTAM supports this adapter, if the migration includes a processor upgrade, a replacement for this adapter must be found. Options include the 3174 with an Asynchronous Emulation Adapter (AEA) providing support for ASCII devices in 3270 emulation mode.
  - ► IBM eNetwork Communications Server for OS/2 (prod #4301114) or Windows NT (prod #4231747) with a Multi-Protocol Adapter (MPA) card can provide support for ASCII devices in 3270 emulation mode.
- Token-Ring Adapter (6139/6140)
- Ethernet Adapter (6135)

- ► The 3174 with a Token-Ring or Ethernet adapter provides direct connection to Token-Ring and Ethernet LANs.
- ► The Open Systems Adapter (OSA) on the CMOS processors provides direct connection to Token-Ring, Ethernet and FDDI LANs. It also supports native ATM connections for VTAM V4R4 and above.
- ► The 3172 Interconnect Controller provides direct attachment to Token-Ring, Ethernet and FDDI LANs.
- ► A 37XX controller with NCP provides SNA connectivity for larger networks over Token-Ring and Frame Relay attachments.

- Workstation Subsystem Controller (6120)
  - ► This adapter is actually a 3174 on a card. Although MVS/VTAM supports this adapter, if the migration includes a processor upgrade, a replacement for this adapter must be found. Options include the use of the integrated console facility of new generation CMOS processors or a channel attached 3174 communications controller.

2 OS/390 VTAM provides SNA Network Interconnection (SNI) capability which is not available in VSE VTAM. SNI:

- Connects multiple independent subarea SNA networks together
- Isolates the topologies of the networks, thus making security easier to enforce and administration simpler
- Permits sessions between any resources in the connected networks, provided the installation has allowed them

SNI is particularly beneficial when you need to link your SNA network with that of another organization such as a business partner or a value-added network supplier.

3 OS/390 VTAM provides high-performance routing (HPR) over APPN connections. HPR has the following benefits:

- More efficient transport over high-speed connections, due to the improved routing and flow control algorithms.
- Nondisruptive session rerouting around failing nodes or links.
- Less processing power required for intermediate nodes on a session path.
- For APPN and HPR networks, there is a range of multi-protocol routers available: the 2210, 2216 and 3746 in increasing order of power and complexity.

4 OS/390 VTAM running in a sysplex provides some functions that are only available in that environment:

- VTAM to VTAM communication using the cross-system coupling facility (XCF) of OS/390. XCF allows all the VTAMs in a sysplex to communicate with each other without requiring any definitions, and without the need to dedicate channel-to-channel connections to SNA traffic.
- Generic resources. This gives improved performance and availability by balancing application load across sysplex images, in a manner transparent to the user.
- Multi-node persistent sessions. This permits sessions to survive a failure in application, VTAM, OS/390 or even a processor in a sysplex

without disruption. Sessions are simply taken over by a new copy of the application running in the same, or a different, processor.

9.1.2.1 Resource Definition

The OS/390 VTAM resource definitions are stored in the VTAMLST data set. Most of the VSE/VTAM resource definitions (B-books), typically stored in the PRD2.CONFIG VSE library, can be moved directly into the VTAMLST data set without modification. There are several items worth noting here:

- If migrating from VSE/VTAM V3R4 or older, MVS and VSE will differ in their use of VTAM buffers. VSE/VTAM V3R4 utilizes the LFBUF pool for I/O and two unique fixed size buffer pools (VFBUF and VPBUF) for pool expansions. VFBUF defines storage for expansion of fixed storage buffer pools and VPBUF for expansion of pageable storage buffer pools. VSE/VTAM V4R2 storage pools and usage was designed to be similar to MVS/VTAM. VFBUF and VPBUF buffer pools were eliminated and the IOBUF buffer pool was added for I/O. IOBUF size should be matched to NCP (UNITSZ) buffer size and tuned for your requirements. All the other OS/390 buffer pool definitions have reasonable defaults and can usually be left alone until it is time for fine tuning.
- If you are converting a Token-Ring ICA attachment to a 3172 or OSA attachment, you will need to replace your Local Area Network (LAN) major node with an External Communication Adapter (XCA) major node to define the connection.
- Where the VTAM definitions refer to a data set, the coding often changes. This mainly occurs in NCP definitions; please refer to 9.2, “ACF/NCP” on page 192 for details.

9.1.2.2 Operation

Most of the OS/390 VTAM console commands will be familiar to the VSE operator. One point of interest is that the DISPLAY, VARY and HALT commands are VTAM commands so that they take the format D NET..., V NET... and so on. On the other hand, START and MODIFY are OS/390 commands so they must refer to the actual name of the VTAM start procedure. Thus if VTAM is started using `S NET28,,,(LIST=S0)` then a subsequent MODIFY command will appear so (for example) `F NET28,TRACE,...`.

9.1.3 Customization and Programming

VTAM tuning can be quite different under VSE and OS/390. Matters such as optimizing I/O across a channel and pacing the flow of traffic are very similar, but OS/390 VTAM uses storage and buffer pools in a completely different fashion than VSE, so this aspect of tuning needs to be reviewed in some detail. Please see the chapter *Tuning VTAM for Your Environment* in the *Network Implementation Guide* for advice on tuning OS/390 VTAM.

9.1.3.1 VTAM Tables

Most of the VTAM tables which are assembled and linked do not differ between OS/390 and VSE, but the use of mode tables needs to be considered.

VSE provides a VTAM mode table called IESINCLM. It is a subset of the default VTAM mode table (ISTINCLM) but also contains some unique entries. IESINCLM is routinely used in VSE systems, but is not provided by OS/390 VTAM. Migration of the table and/or unique entries may be required. Refer to the *VSE/ESA Networking Support* manual for more information on IESINCLM.

9.1.3.2 Programming

Any coding done under VSE, such as VTAM exits, will almost certainly need rewriting (and will certainly need re-linking) for the OS/390 environment. Also, some VTAM exit routines may be implemented differently under VSE and OS/390. User-written VTAM programs and exits must be reviewed carefully for compatibility. Please refer to the chapter *Operating System Facilities* in the *VTAM Programming* manual.

9.1.4 Network Configuration

It is not unusual for a migrating customer to require multiple host access to the SNA network. Multiple host configurations can serve to reduce or even eliminate service disruptions during the migration process. Multiple host configurations can provide for both interactive and batch traffic between all connected images. This means that any terminal, regardless of to which host it is physically connected, can access any VTAM application in the network, regardless of in which host the application resides. In addition, SNANJE connections can be set up between JES and PNET. For a migrating customer, these capabilities can be particularly useful, providing simultaneous access to both old and new systems as well as file transfer capability between them.

VTAM provides for two different multiple host networking architectures, traditional SNA subarea (cross domain) and/or APPN (Advanced Peer-to-Peer Networking). VSE/VTAM V3R4 is limited to SNA subarea because it does not provide full APPN capability. The MultiDomain package of VSE/VTAM V4R2 is required to support multiple host connectivity in an SNA subarea network. VSE VTAM V4R2 provides APPN capability at all three functional levels, but with some limitations in the Client/Server and MultiDomain packages. OS/390 VTAM is full featured and provides both SNA subarea and APPN capabilities. Implementation requires careful planning and solid networking skills. Some points to consider in network design are:

- Number of hosts
- Number of applications
- Number of terminal resources
- Number of NCPs
- Traffic patterns
- Future growth
- Backup/redundancy requirements
- Bandwidth requirements
- Operation/management
- Security
- Cost

Concepts and technical explanations, as well as implementation techniques for SNA networking in both subarea and APPN designs, can be found in the following manuals:

- *IBM Network Products Implementation Guide*, GG24-3649
- *VTAM Network Implementation Guide*, GC31-8370

9.2 ACF/NCP

ACF/NCP in a 37XX controller may itself be completely independent of the operating system in the host, but the generation and loading of such an NCP is very much dependent on the operating system.

9.2.1 Product Installation

Differences in the installation procedures of NCP for VSE and OS/390 are basically the same as those for VTAM. The main steps required to implement an NCP under OS/390 are:

- Allocate data sets for use by NCP and the generation process.

  The generation process requires three input PDSs and a number of work files in order to process the source statements. The three PDSs, into which the SSP and NCP modules are installed, are:

  – The data set containing the ACF/SSP modules, which includes the load and dump utilities as well as the NDF generation programs. This is also referred to in the VTAM start procedure (STEPLIB) because VTAM uses it to load the NCP.
  – The data set containing the ACF/NCP macro statements used to assemble the NCP object modules which are created during the NDF process. It is referred to as SYSLIB in the generation procedure. It is a source library and therefore has DCB=(RECFM=FB,LRECL=80) with a suitable block size.
  – The data set containing the ACF/NCP modules used in link-editing the NCP into its final home. As a load module library it has DCB=(RECFM=U).

  You must also allocate the data set into which the finished NCP will go (NCPLOAD in our example VTAM start procedure).

- Install the SSP and NCP products to the libraries, using SMP/E. Current levels of ACF/NCP and ACF/SSP can be included in the OS/390 SystemPac with the initial installation tasks already completed.

- Run the program generation procedure.

9.2.2 Program Generation

Generation under VSE and OS/390 is done using NDF. ACF/SSP beginning with Version 3 includes the NCP/EP Definition Facility (NDF), a program used in generating an NCP and/or EP load module. Compared to the old generation processes, NDF can do a program generation four to eight times faster. There is also a time-saving FASTRUN option in NDF which can be used to validate the NCP/EP definition macro coding without invoking the generation process. The control program generation using NDF under VSE is a six step process, while under OS/390 it is a two step process. These steps are submitted as one single job and no operator/programmer intervention is required between the steps. More information on NDF (including samples) can be found in the *NCP, SSP, and EP Generation and Loading Guide*.

There are differences in the actual coding of the NCP between VSE and OS/390, because the definition statements refer to operating system-dependent facilities. In particular:

- On the PCCU statement there are DUMPDS, MDUMPDS and CDUMPDS keywords which refer to various data sets which will contain NCP dumps. Code the names of the DD statements in the VTAM procedure which will point to the actual data sets.
- On the BUILD statement, the LOADLIB keyword specifies the DD name of the data set which VTAM will use to find the NCP when the time comes to load it. The name you code must be in the VTAM start procedure (NCPLOAD in the example).

9.2.3 Backlevel Hardware Support

Current releases of NCP do not support older 37XX hardware. Many of these boxes are still in service, so if back level 37XX hardware support is required the following NCP versions remain orderable for both VSE and OS/390:

- NCP V4R3.1 (5668-854) - 3725 support
- NCP V5R4.0 (5668-738) - 3720 support

You should be aware, however, that if you are using NCP V4R2 (or below) on a 3725 or 3720, the upgrade to a supported version will use a lot more NCP storage. NCP V4R3 introduced support for independent LUs for the first time, and the resulting increase in module size (even if ILUs are not used) can add hundreds of Kilobytes to your requirements.

You should also be aware that neither the 3725 nor the 3720 can support APPN or HPR.

9.3 BTAM

9.3.1 Product Installation

Differences in the installation procedures of BTAM for VSE and OS/390 are basically the same as those for VTAM and NCP. The VSE BTAM-ES product (5746-RC5) and the OS/390 BTAM/SP product (5665-279) are very similar in the MACRO names they use and the function they provide. The differences are in the parameters they require on the MACROS. For further information refer to *BTAM-ES Programming Reference*, SC38-0293 for VSE and *BTAM/SP*, SC27-0604 for OS/390.

9.3.2 Usage

CICS is the most common user of BTAM. Although BTAM continues to be supported by both VSE and OS/390, note that there is no support for devices and controllers accessed using BTAM in any CICS version after CICS/MVS Version 2 Release 1.2. The recommendation here is to migrate these devices to VTAM.

9.4 Migrating TCP/IP

TCP/IP provides the ability to merge differing physical networks while giving users a common suite of functions. It allows interoperability between equipment supplied by multiple vendors on multiple platforms. TCP/IP is ***the*** protocol in an open networking world including the Internet.

One of the reasons why TCP/IP is so popular is that there are many simple and useful standard applications available. The TCP/IP on VSE/ESA for example provides standard applications such as Telnet, FTP, LPR/LPD and the HTTP Web server.

Using these standard TCP/IP applications and standard TCP/IP APIs for user written applications also allows an easy migration from one TCP/IP platform to another.

The VSE products on which the following migration discussion is based are TCP/IP for VSE/ESA 1.3 (which comes with VSE/ESA 2.3) and TCP/IP for VSE 1.3 from Connectivity Systems Inc..

The migration target system is OS/390 R3 with TCP/IP OpenEdition, OS/390 V2R4 with TCP/IP UNIX Services or the soon to come OS/390 V2R5 with eNetwork Communications Server for OS/390 V2R5 which includes TCP/IP.

All the standard TCP/IP applications such as Telnet, FTP, LPR/LPD or HTTP that are available with TCP/IP on VSE are also part of TCP/IP on OS/390. Since, generally speaking, TCP/IP functions on VSE are a subset of those on OS/390, migration from TCP/IP for VSE to TCP/IP on OS/390 can be accomplished without loss of functionality in most cases.

The topics you should look into if you want to migrate from a TCP/IP for VSE environment to TCP/IP on OS/390 include:

- network attachments and definitions required to communicate with other TCP/IP systems
- TCP/IP configuration
- TCP/IP related user data
- your TCP/IP batch jobs
- your own TCP/IP based application programs
- TCP/IP security.

In the next sections each of these topics will be addressed at a general level. No detailed checklist or migration guidelines are provided. This would go beyond the scope of this documentation. The purpose of this discussion is to make you aware of the areas which need to be studied in detail when you intend to migrate.

9.4.1 Network Definitions

The connectivity/network attachments supported by TCP/IP on OS/390 is a superset of what is supported by TCP/IP for VSE. This is why your networking environment basically does not have to be changed. However, since OS/390 TCP/IP supports more communication interfaces, your network can be changed or extended for example by an IBM 3746 attachment.

As on VSE/ESA, all the network attachments to be used by TCP/IP have to be defined to the OS/390 operating system first. The network attachments (for example CTCA, IBM3172, OSA-2) need to be defined before they can be specified in the TCP/IP configuration.

9.4.2 TCP/IP Configuration

First of all, configure the UNIX System Services (part of the OS/390 base product) in order to enable TCP/IP on OS/390. As a second step you will have to customize TCP/IP on OS/390.

9.4.2.1 TCP/IP Customization

On VSE/ESA, almost all TCP/IP customization information is located in one file, the IPINIT.L initialization file. It contains all the required TCP/IP definitions such as the physical network, links, routing tables, user IDs, and TCP/IP daemons.

On OS/390, the TCP/IP definitions are located in several data sets. This is why you have to customize several data sets in order to migrate your VSE/ESA TCP/IP definitions. Mainly, you have to provide the:

- TCPIP.PROFILE (TCP/IP definitions for the physical network, network routing, TCP/IP stack and so on)
- TCPIP.DATA (parameters for all TCP/IP server and client functions)
- depending on your requirement, some other configuration data sets such as HOSTS.LOCAL (host name to IP address mapping) and other etc files have to be set up.

9.4.2.2 TCP/IP Standard Applications

All TCP/IP for VSE standard applications such as Telnet, FTP or LPR/LPD, are also available with TCP/IP on OS/390. The HTTP server on OS/390 is provided by the Domino Go Web Server or the ICSS (Internet Connection Secure Server). Therefore, migrating to OS/390 doesn′t restrict the TCP/IP server and client functionality and can be done with low effort. However, OS/390 users can make use of additional TCP/IP functions such as REXEC, DCE or SMTP.

9.4.3 TCP/IP Related User Data

Don′t forget to move your data files that are exclusively used/accessed through TCP/IP. If you have, for example, set up VSE/ESA as a Web server, consider to move the Web server data such as HTML documents, and graphics that are stored in VSE library members or VSAM files. These files should be moved into OS/390 HFS (Hierarchical Filesystem) data sets. Transferring the files could, for example, be done using the FTP function.

9.4.4 TCP/IP Batch Jobs

If you are using batch jobs on VSE to automate TCP/IP client commands such as LPR printing, FTP to automatically transfer files or the Telnet client to access remote TCP/IP systems, take into account that you will have to migrate these batch jobs as well. For the OS/390 system, you have to convert your batch jobs to CLISTs, REXX EXECs or UNIX shell scripts.

9.4.5 User Written TCP/IP Applications

If you have many user written TCP/IP applications on your VSE/ESA system, the migration effort can be considerably high depending on the TCP/IP interface that has been used.

9.4.5.1 TCP/IP Applications using the Sockets API for Assembler

VSE/ESA applications based on the SOCKET Assembler macro cannot be used on an OS/390 system. They have to be recoded for the OS/390 TCP/IP.

9.4.5.2 TCP/IP Applications using the Preprocessor API

The HLL preprocessor API which is available on VSE/ESA for PL/I, Assembler and COBOL is not compatible with the OS/390 TCP/IP interfaces. Therefore, these programs have to be recoded for the OS/390 system as well.

9.4.5.3 TCP/IP Applications using the BSD/C Sockets

The BSD (″Berkeley″) C sockets interface on VSE/ESA is almost compatible to the C socket API on OS/390. Only some additional (proprietary) functions or parameters of the BSD/C interface are not supported by TCP/IP on OS/390. This is why VSE/ESA TCP/IP applications based on the BSD/C sockets usually can be migrated to OS/390 with only minor code changes.

9.4.5.4 TCP/IP Applications using the LE/VSE C Socket API

It is highly recommended to use the IBM C for VSE/ESA compiler, the IBM Language Environment for VSE/ESA (LE/VSE) C run-time environment and the **LE/VSE C socket interface** to write TCP/IP applications on VSE/ESA. These are compatible with the OS/390 X/Open (XPG4.2) compliant socket interfaces. This assures the maximum in compatibility and portability for cross platform development. In this case, migrating the applications is just a matter of relinking them on the OS/390 system. More information about sockets programming can be found in the *TCP/IP for VSE/ESA User′s Guide*, SC33-6601.

The VSE applications don′t have to be necessarily C program since you can use the LE/VSE C socket API also from within other languages using the ILC (InterLanguage Communication). This is described in the book *Writing Interlanguage Communication Applications*, SC33-6686.

9.4.5.5 CGI Programs

If you are running VSE/ESA as a Web server and therefore have implemented CGI (Common Gateway Interface) programs on the VSE system, all these programs have to be rewritten on OS/390 since the CGI interface on OS/390 is totally different to the one on VSE/ESA.

9.4.6 Security

Security is an important consideration for an OS/390 system, especially if it′s connected to large TCP/IP networks or even the Internet. TCP/IP on OS/390 has some built-in internal security mechanism and relies on the services of an external security manager such as IBM Resource Access Control Facility (RACF).

Basic TCP/IP security definitions on VSE (such as user ID/password) can be easily defined in RACF for the OS/390 system. If you have implemented your own security exit on VSE, similar exits can be written for the FTP server function on OS/390. Furthermore, RACF can be used to protect whole libraries or single resources from unauthorized TCP/IP access.

Additionally, the OS/390 system can be run as a firewall to secure the system against users coming through the TCP/IP network.

Generally, you can achieve a higher level of security on the OS/390 system which, of course requires a little more effort to set it up.

9.4.7 Bibliography

VSE/ESA

SC33-6601	*TCP/IP for VSE/ESA User's Guide*
SG24-2041	*The Native TCP/IP Solution for VSE*
SG24-2040	*VSE/ESA as a Web Server*
SC33-6686	*Writing Interlanguage Communication Applications*

OS/390

SC28-1890	*OS/390 OpenEdition MVS Planning*
SC28-1906	*OS/390 OpenEdition MVS Communications Server Guide*
GC28-1920	*OS/390 Security Server (RACF) Planning: Installation and Migration*
SC28-1915	*OS/390 Security Server (RACF) Security Administrator*

9.5 MQSeries

MQSeries represents a family of products which enable applications to use message queuing to participate in message-driven processing. With message-driven processing, applications can communicate across the same or different platforms, by using the appropriate message queuing software products. For example, VSE/ESA and OS/400 applications can communicate through MQSeries for VSE/ESA and MQSeries for OS/400 respectively. With MQSeries products, all applications use the same kinds of message headers. Communications protocols are hidden from the application.

MQSeries products implement a common application programming interface, the message queue interface (MQI), that is used on whatever platform the applications run. The calls made by the applications and the messages they exchange are common. This makes it much easier to write and maintain applications than using traditional methods. It also makes it easier to migrate message queuing applications from one platform to another.

The current releases on which the following migration discussion is based, are MQSeries for VSE/ESA Version 1.4 and MQSeries for MVS/ESA Version 1.2.

MQSeries for VSE/ESA 1.4 is a so-called level 1 product whereas the OS/390 version is a level 2 product. Level 2 products offer functions and facilities which extend those available in level 1 products. Since, generally speaking, level 1 functions are a subset of level 2 functions migration from MQSeries for VSE/ESA to MQSeries for MVS/ESA can be accomplished without loss of functionality in most cases.

The topics you should look into if you want to migrate from a VSE/ESA based MQSeries environment to one based on an OS/390 system include:

- the setup for MQSeries within your operating system environment
- the networking definitions required to communicate with other MQSeries systems
- the definition of your MQSeries objects such as queues and channels
- the operating facilities offered to monitor MQSeries activities and diagnose and fix problems if they occur

- your MQSeries based applications.

In the next sections each of these topics will be addressed at a general level. No detailed checklists or migration guidelines are given. This would go beyond the scope of this chapter. The purpose of this discussion is to make you aware of the areas which need to be studied in detail when you intend to migrate.

Only straight forward migration is considered. MQSeries applications under VSE/ESA always:

- run under CICS for VSE/ESA
- are coded in COBOL and compiled with COBOL for VSE
- communicate with other systems through SNA/LU6.2.

Under VSE/ESA you do not have any other options.

The basic assumption is that you migrate to MQSeries for MVS/ESA as directly as possible:

- using CICS/ESA to run your migrated MQSeries applications
- staying with COBOL as the programming language
- still using SNA/LU 6.2 connections to communicate with other MQSeries systems.

If you want to make use of the additional facilities of MQSeries for MVS/ESA, such as IMS support or C language support or TCP/IP connectivity, this would have to be done in a separate, second step not discussed here.

9.5.1 MQSeries in Your Operating System Environment

You will have to verify that you have the necessary software prerequisites on your OS/390 system to run the migrated applications, install the MQSeries for MVS/ESA product, verify that you have space for the required data sets and define them on your OS/390 system. These points will be addressed below.

9.5.1.1 Prerequisites

MQSeries for VSE/ESA V1.4 (product number 5787-ECX) is an MSHP-installable licensed program. It operates with CICS for VSE/ESA, providing coordination between MQSeries and CICS resources by allowing CICS for VSE/ESA transactions to issue MQSeries (MQI) calls.

MQSeries for VSE/ESA does not run in a VSE partition. It executes under control of CICS for VSE/ESA.

It supports communication with other MQSeries products through SNA/LU 6.2.

The following list shows the required products with product numbers:

- VSE/ESA Version 1.4 (5750-ACD) in ESA mode, with:
  - CICS for VSE/ESA Version 2.3 (5686-026)
  - IBM Language Environment (LE) for VSE Runtime Library (5686-067)
  - ACF/VTAM for VSE/ESA V3.4 (5666-363)

  or

- VSE/ESA Version 2.1 (5690-VSE), with:
  - CICS for VSE/ESA Version 2.3 (5686-026)
  - IBM Language Environment (LE) for VSE Runtime Library (5686-067)
  - ACF/VTAM for VSE/ESA V4.2 (5686-065)

Applications using MQSeries must be written in COBOL using

- IBM COBOL for VSE (5686-068)

MQSeries for MVS/ESA V1.2 is a licensed program. It operates in the MVS/ESA environment as a separate MVS subsystem. You need to have a separate address space to run MQSeries with storage space allocated both above and below the 16MB line.

MQSeries for MVS/ESA can be accessed from CICS/ESA, CICS/MVS, IMS/ESA, batch MVS/ESA environments, and TSO/E.

To interoperate with other systems, the attachment can be via either TCP/IP or SNA/LU 6.2 APPC protocols. A complete installation needs at least one other MQSeries queue manager (on a similar or a different platform) to communicate with MQSeries for MVS/ESA. In addition, MQSeries for MVS/ESA can support MQSeries clients using the optional Client Attachment feature.

The following list show the required products with product numbers:

- MQSeries for MVS/ESA (5695-137)
- MVS/ESA 4.3, or later (5695-047 (JES2) or 5695-048 (JES3)) or
  MVS/ESA 5.1, or later (5655-068 (JES2) or 5655-069 (JES3))
- SMP/E 1.8 (5668-949)
- DFSMS/MVS binder utility (5695-DF1)
- DFP 3.1, or later (5665-XA3)
- RACF 2.1 (5695-039)
- ISPF/PDF, or later (5685-054)
- TSO/E 2.0, or later (5685-025) for batch access
- CICS/MVS 2.1.2 (5665-403) or
  CICS/ESA (R) 3.2.1, or later (5685-083) or
  CICS/ESA 3.3, or later (5685-083) for CICS access
- IMS/ESA 3.1, or later (5665-409) or
  IMS/ESA 5.1, or later (5695-176)
- SAA AD/Cycle LE/370 (5688-198)
- ACF/VTAM 3.4.1 (5685-085)
- IBM TCP/IP 3.1 (5655-HAL) or
  IBM TCP/IP 3.2 (5655-HAL)

The following languages and compilers are supported for MQSeries applications:

- Assembler
  - Assembler H (5668-962)
  - IBM High level assembler MVS (5696-234)
- COBOL
  - VS COBOL II (5668-958)
  - IBM COBOL for MVS & VM Release 2 (5688-197)
- C
  - C/370 Release 2.1.0 (with APAR UN37741) (5688-187)
  - IBM SAA AD/Cycle C/370 (5688-216)
- PL/I
  - OS PL/I Optimizing Compiler (5668-910)
  - SAA AD/Cycle PL/I Compiler (5688-235)
- Java
  - OS/390 Java Development Kit 1.0.2 available at *http://ncc.hursley.ibm.com/javainfo/download/index.html*

9.5.1.2 Installation and Customization

The most important prerequisite for migrating from VSE/ESA is that you have set up a CICS/ESA and the COBOL compiler and runtime environment.

After you have verified that your OS/390 system fulfills the prerequisites you can install the MQSeries for MVS/ESA code. Follow the instructions in the MQSeries for MVS/ESA Program Directory. They include not only details of the installation process but also information about any necessary prerequisite products and their service or maintenance levels.

SMP/E, used for installation on the MVS/ESA platform, validates the service levels, prerequisite and corequisite products, and maintains the SMP/E history records to record the installation of MQSeries for MVS/ESA. It loads the MQSeries for MVS/ESA libraries and checks that the loads have been successful.

You now have to customize the product to your own requirements. Customization tasks include:

- define the MQSeries for MVS/ESA subsystem to MVS
- authorize the MQSeries for MVS/ESA load libraries
- include the MQSeries for MVS/ESA load library in the link list
- include the MQSeries for MVS/ESA dump formatting member
- update the MVS/ESA Program Properties Table (PPT)
- create procedures for the MQSeries for MVS/ESA subsystem
- tailor your security procedures
- customize the initialization files
- create the bootstrap and log data sets
- define your page sets
- tailor your logging environment

- install required CICS and IMS adapters
- define queues
- set up distributed queuing

Customization is described in detail in the *MQSeries for MVS/ESA System Management Guide*. Only the steps which are related to migration are discussed in some more detail in the following sections.

After the installation and customization has been completed, you can use an installation verification program (IVP) supplied with the product to verify that the installation has been completed successfully. Details of the IVP and customization are given in the *MQSeries for MVS/ESA System Management Guide*.

9.5.1.3 CICS Considerations

The CICS execution environment is the only environment for VSE/ESA based MQSeries applications. Naturally it will, therefore, be the main migration target environment.

There is a major difference between MQSeries under VSE/ESA and under OS/390:

- Under VSE/ESA MQSeries itself and MQSeries applications run under CICS. Therefore, MQSeries resources (such as programs, queues) must be defined in CICS/VSE. Access to both MQSeries and CICS resources from an application (which includes both CICS commands and MQI calls) does not require additional installation or customization effort.
- Under OS/390 MQSeries runs in an address space independent of CICS. To allow CICS applications to access MQSeries resources (through API calls) the CICS adapter must be installed and customized in the respective CICS region. The CICS adapter connects a CICS subsystem to an MQSeries subsystem, enabling CICS application programs to participate in message-driven processing.

  The CICS adapter provides two main facilities:

  – a set of control functions for use by system programmers and administrators to manage the adapter.

    Control functions let you manage the connections between CICS and MQSeries dynamically. They may be invoked using the CICS adapter panels, from the command line, or from a CICS application. You can use the adapter′s control function to:

    - start, stop and modify a connection to a queue manager
    - display the current status and statistics of a connection
    - start and stop an instance of the task initiator transaction, CKTI
    - display details of the current instances of CKTI
    - display details of the CICS tasks currently using the adapter.

  – MQI support for CICS applications.

    For performance, the CICS adapter can handle up to eight MQI calls concurrently. For transaction integrity, the adapter fully supports syncpointing under the control of the CICS syncpoint manager. The adapter also supports security checking of MQSeries resources when used with an appropriate security management product, such as RACF. The adapter provides high availability with automatic reconnection after

an MQSeries termination, and automatic resource resynchronization after a restart.

The CICS adapter is supplied with MQSeries as the CICS transaction CKQC.

9.5.1.4 Data Sets

MQSeries for VSE/ESA uses the following data sets:

- the System Setup file: a VSAM ESDS file containing system setup information. It is only used once to initialize the System Configuration file.
- the System Configuration file: a VSAM KSDS file. It initially contains system definitions, messages, names of MQSeries maps and programs and so on. It will be updated whenever the user defines additional MQSeries objects such as queues and channels.
- Queue data sets: VSAM KSDS files to hold messages.

MQSeries for MVS/ESA uses the following data sets:

- Page sets to store messages and definitions.

  A page set is a linear VSAM data set that has been formatted for use by MQSeries for MVS/ESA. Each page set is identified by a page set ID (PSID), an integer in the range 00 through 99. In particular, MQSeries for MVS/ESA uses page set 00 to store queue definitions and other important information relevant to the queue manager.

- Log data sets to log data and events.

  MQSeries for MVS/ESA records all persistent messages in the active log. When the active log is full, MQSeries for MVS/ESA switches to the next available log data set. Each active log data set is a single-volume, single-extent VSAM entry-sequenced data set (ESDS).

  If archiving has been switched on during customization, MQSeries for MVSE/ESA copies the contents of a full active log to an archive log when it switches logs. The archive logs can be a data set on a direct access storage device (DASD) or on magnetic tape. The archive log consists of up to 1000 sequential data sets. Each data set can be cataloged using the Integrated Catalog Facility (ICF).

  MQSeries for MVS/ESA allows you to have either single logging or dual logging.

- The boot strap data set is a VSAM key-sequenced data set (KSDS) that holds information needed by MQSeries for MVS/ESA. It contains an inventory of all active and archived log data sets known to MQSeries for MVS/ESA.

No migration of the data sets used by MQSeries for VSE/ESA to those used by MQSeries for MVS/ESA is supported.

To set up MQSeries for MVS/ESA you need to define and populate the required data set from scratch.

You can use the information on the size of your VSE/ESA data sets to estimate the space required under OS/390.

No facility exists to move messages from a VSE/ESA queue to a queue under OS/390. Therefore, you should make sure that all your queue entries have been processed under VSE/ESA before you switch to the OS/390 system.

9.5.2 Networking Definitions

You will have some other systems with an MQSeries product installed which are connected to your VSE/ESA to allow MQSeries applications to communicate. When you migrate your VSE/ESA to OS/390 you will have to re-establish connection to those systems.

In this section we will discuss the networking considerations. There are also MQSeries definitions which refer to remote MQSeries systems. These are discussed in the next section.

We assume again, that we only talk about straight forward migration. In this context this means that we only need to re-establish the SNA/LU6.2 connections. MQSeries for MVS/ESA also supports TCP/IP as a communications protocol. But since this is not supported for MQSeries for VSE/ESA, it will not be part of the direct migration effort.

There are two parts to consider:

1. establish SNA connection between OS/390 and the other systems running MQSeries
2. check and, if necessary, modify the SNA definitions in the remote systems to connect to OS/390 rather than to VSE/ESA.

On the OS/390 side you will have to set up VTAM definitions equivalent to the definitions you made in VSE/ESA. You may be able to modify and use the VSE/ESA B-books containing the equivalent definitions.

What you need to do on the remote systems depends on what has changed. In an ideal case: If you could reuse the same communications hardware, such as communications controllers and so on, and define the same names for VTAM resources under OS/390 which you used under VSE/ESA, you would not have to change anything. More realistically, however, you will probably at least have to change remote addresses (such as MAC addresses) and remote names (such as partner LU names) on all the systems you want to connect to OS/390.

9.5.3 Defining MQSeries Object and Operating

Under VSE/ESA you will have defined the following MQSeries system elements:

1. the queue manager
2. message queues
3. channels.

These elements and their properties are defined through CICS panels. No command interface exists for MQSeries for VSE/ESA.

The definitions are stored in the MQSeries configuration file.

In MQSeries for MVS/ESA, there are six different types of object:

1. queue managers
2. queues
3. channels

Note: If you are using CICS for distributed queuing, channels are not objects, and cannot be manipulated using MQSeries commands (see below).

4. namelists
5. process definitions
6. storage classes

These objects can be manipulated, that is, defined, deleted, changed, by the MQSeries commands.

Commands can be issued from:

- the initialization input data sets
- the MVS console
- the system-command input queue
- the COMMAND function of the CSQUTIL utility
- the operations and control panels using ISPF.

When you migrate to a CICS/ESA environment, your channels are special. They must be handled from the CICS region rather than through the MQSeries region: You monitor and control the channels to remote queue managers with the DQM (Distributed Queue Management) panels. Each MVS queue manager has a set of DQM CICS transactions for controlling interconnections to compatible remote queue managers using CICS ISC facilities.

The DQM channel control function (CCF) provides the administration and control of message channels. The channel definition file (CDF):

- is a VSAM file
- is indexed on channel name
- holds channel definitions
- must be available to the CICS regions in which the channel control program runs and where the MQSeries Channel adapters run.

You use channel definition panels to:

- create, copy, display, alter, find and delete channel definitions
- start channels, reset channel sequence numbers, stop channels and so on
- display status information about channels.

As you can see, the VSE/ESA related MQSeries objects (and more) exist also for MQSeries for MVS/ESA. Similarly, most of the parameters used to define these objects are identical or have an equivalent which allows you to reproduce or extend the definition you used under MQSeries for VSE/ESA.

But you can also see that the way the objects are defined is different in the OS/390 environment. There is no way to directly migrate the VSE/ESA definitions.

With the MQPUTIL program shipped with MQSeries for VSE.ESA V.1.4 you can print a listing with the MQSeries system, queues, and channels defined under VSE/ESA. You have to use this listing to recreate equivalent definitions for MQSeries for MVS/ESA. You will have to:

- print the listing under VSE/ESA
- check the DEFINE commands in the *MQSeries Command Reference*, SC33-1369 to find matching parameters to recreate the queue manager and queue definitions
- decide on the environment from which you want to issue the commands
- run the commands

- for channel definitions under CICS/ESA find matching channel attributes in *MQSeries Distributed Queuing Guide*, SC33-1139
- define the channels using the CKMC CICS transaction.

Note: Make sure that you use the same queue names in your OS/390 related definitions which you used under VSE/ESA whenever possible. This will minimize application changes.

Operating MQSeries for VSE/ESA includes:

- initializing the MQSeries system and shutting it down
- starting and stopping queues
- opening, resetting and closing channels.

Monitoring offers functions to:

- monitor channel and queue activities
- browse queues.

Operating and monitoring facilities are provided through the MQSeries menu under CICS for VSE/ESA.

For MQSeries for MVS/ESA similar functions are provided:

- for system and queue definitions through MQSeries commands
- for message channels under CICS through the DQM panels.

There is no direct migration support. You will have to make sure your operational staff is trained to use the MQSeries for MVS/ESA facilities. You can set up some procedures to automate these tasks.

9.5.4 MQSeries-based Applications

Applications using MQSeries for VSE/ESA are coded in COBOL for VSE and run under CICS for VSE/ESA. They use the (MQSeries for VSE/ESA specific subset of) MQI.

The first two application migration steps are independent of MQSeries:

1. you have to migrate a program written in COBOL for VSE to OS/390. The best choice is to migrate the application to IBM COBOL for MVS & VM Release 2.
2. you have to migrate a program written to run under CICS for VSE/ESA to run under CICS/ESA.

For some details on these steps refer to the respective chapters in this book.

The next step is to consider the Message Queueing Interface (MQI). The effort required for this step should be small since on a high level the MQI is identical for all platforms.

You should use the*MQSeries Application Programming Reference*, SC33-1673. This book gives a full description of the MQSeries programming interface for MQSeries for MVS/ESA. You should check that the MQI elements you used in your VSE/ESA based applications are identical with the ones under OS/390. Minor adaptations may be required.

As a final step you need to compile and link your applications under OS/390.

No special considerations apply to the compile and link step under VSE/ESA except that the product library which contains MQSeries for VSE/ESA has to be in the library chain during compilation.

You have to set up jobs to compile and link the program under OS/390. General considerations how to do this for a CICS COBOL application are described in the respective chapters in this book. In addition the following applies to MSQeries applications:

- include in the SYSLIB statement of the compilation statements that make the product data definition files available to the compiler. For COBOL the data definitions are supplied in the library thlqual.SCSQCOBC.
- in your link-edit JCL, the MQSeries for MVS/ESA CICS stub program (CSQCSTUB) must be included. The stub is language independent and is supplied in library thlqual.SCSQLOAD.

For details please refer to *MQSeries Application Programming Guide*, SC33-0807.

9.5.5 Bibliography

For MQSeries for VSE/ESA:

- *MQSeries for VSE/ESA Version 1 Release 4 Users Guide*, SC33-1142

For MQSeries for MVS/ESA:

- *MQSeries Application Programming Guide*, SC33-0807
- *MQSeries Planning Guide*, GC33-1349
- *MQSeries Application Programming Reference Summary*, SX33-6095
- *MQSeries Intercommunication*, SC33-1872
- *MQSeries Command Reference*, SC33-1369
- *MQSeries Programmable System Management*, SC33-1482
- *MQSeries Application Programming Reference*, SC33-1673
- *MQSeries for MVS/ESA Licensed Program Specifications*, GC33-1350
- *MQSeries for MVS/ESA System Management Guide*, SC33-0806
- *MQSeries for MVS/ESA Problem Determination Guide*, GC33-0808
- *MQSeries for MVS/ESA Messages and Codes*, GC33-0819
- *MQSeries Clients*, GC33-1632

Chapter 10. POWER and JES2

10.1 JES2 Introduction

VSE uses POWER as a spooling system. MVS uses either JES2 or JES3 as spooling systems. This chapter only addresses migrations from POWER to JES2.

While POWER and JES2 are similar in their overall function, they are very different in design and specific implementations. Because of these differences in processing spool output, you may have to redesign some of your output procedures when migrating to MVS.

No attempt is made here to discuss the many advanced functions available in JES2. You should learn them from JES2 education classes and the JES2 publication library. What will be addressed will be a comparison of POWER functions and their JES2 equivalents as appropriate.

The chapter is divided into four sections:

1. The first section introduces the major POWER and JES2 functional differences, including migration considerations.
2. The second section describes significant tasks required to implement JES2.
3. The third section compares the functions and capacity of POWER and JES2 in more detail.
4. The fourth section shows a detailed mapping of POWER parameters, exits and commands to their JES2 counterparts.

10.1.1 Major Differences

There are some major POWER unique functions, which, if used, will have to be accomplished in some other way in MVS. The following lists these functions and possible MVS alternate suggestions.

10.1.1.1 KEEP Disposition for Pre-Execution Jobs

In POWER, the user can specify that an input job be kept. Thus after the job is executed, it still remains in the POWER spool, for submission over and over.

In JES2 there is no KEEP disposition for spooled input jobs, as there is for SYSOUT data sets. Possible solutions in MVS to accomplish retaining submitted jobs include:

- Append a jobstep to the end of these jobs to resubmit themselves in TYPRUN=HOLD condition.
- Use a standard automated job scheduling package, such as OPC/ESA.
- With SDSF and ISPF, you can edit the job's JCL and resubmit the job, as long as it has not been purged from the JES2 spool. (Keep some output around so it is not purged.)

10.1.1.2 Time Event Scheduling for Jobs

POWER supports the scheduling of job submission based on a one-time or repetitive schedule such as daily, weekly on a given day and time, and so on.

JES2 has a primitive ″automatic command scheduling″ facility, but you will probably need an automated job scheduling package in OS/390 to do the same things you were doing with POWER.

10.1.1.3 Tape Spooling

Spooling to tape is usually done for very large SYSOUT data sets where there is limited spool space or where the data is to be archived. JES2 does not provide direct spooling to tape.

For very large output files, or to archive critical output, you can use the JES2 Spool Offload facility or some spool archiving package such as RMDS or R/DARS.

For data exchange to offline devices or programs, use the IBM supplied external writer (XWTR procedure) to write spooled data to tape.

Another possible solution in MVS to accomplish tape spooling is instead of allocating a file to SYSOUT, you can specify a unit on the DD statement to direct it to tape (for example, UNIT = TAPE). Then use the MVS utility IEBGENER to print the tape with the SYSUT2 DD statement specifying the address of the desired printer (for example, UNIT=00E). Note: The printer has to be ″drained″ from JES2 before it can be allocated to the IEBGENER utility program.

Use of the above procedure retains (KEEPs) the output on tape as long as the installation has a retention requirement.

10.1.1.4 Printer Forms Alignment via PSETUP

The POWER ″PSETUP″ function allowed the operator to print a couple of pages of X′s in order for the operator to line up the form. JES2 does not provide this same function. Here are some alternatives:

- With JES2, you can start the printing, and if an alignment adjustment is needed, interrupt the printer, do the alignment, and then issue the JES2 Backspace command. ″$B PRTnnnn,D″ (where nnnn is the printer number) will logically backspace to the beginning of the data set. Printing resumes at the beginning of the data set. (This assumes the data set being restarted is large enough not to be completely printed before it is backspaced.)
- Another technique is to have some dummy reports kept on spool for each form that needs alignment. Release the dummy output group and cause it to be selected first so you can line up the printer.

10.1.1.5 Separator Page Difference

Operators may be using the VSE separator pages to separate individual copies of reports. JES2 handles separator pages differently than VSE. Under VSE, when a printout has multiple copies, separator pages are printed between each copy. With JES2, you can use “output group” separator pages (one at the beginning and one at the end) “data set” separators, and “data set copy” separators between each copy.

For output group separators, specify SEP=YES on the JES2 PRT(nnn) initialization statement. You can use the IBM-supplied separator pages, or you

can write your own using JES2 exit 1. (For PSF controlled printers, use PSF exits APSUX01 and APSUX02.)

For separators between individual data sets or data set copies, you must specify SEPDS=YES on the PRT(nnnn) statement and provide a JES2 Exit 15 for JES2 controlled printers. (For PSF controlled printers, use PSF exit APSUX03.)

10.1.1.6 End-of-page Sensing

POWER, by storing a carriage control tape image, allows a program to ″sense″ channel 12 or end-of-page. This is not supported in JES2. Line count logic must be substituted. You can change the application to count lines under VSE prior to the MVS conversion.

10.1.1.7 FCB Incompatibilities

POWER permits the ″channel one″ position to be on any line of the page. JES2 requires Channel 1 to be on Line 1 for all printers controlled by JES2.

This restriction is in place because JES2 uses a skip-to-channel-1 to reposition the printer to the top of forms when it goes through device setup.

Also see 10.3.4.8, “FCB Naming Differences” on page 217.

10.1.1.8 Other Differences

Incompatibilities such as JCL and Operator Commands are covered in other chapters:

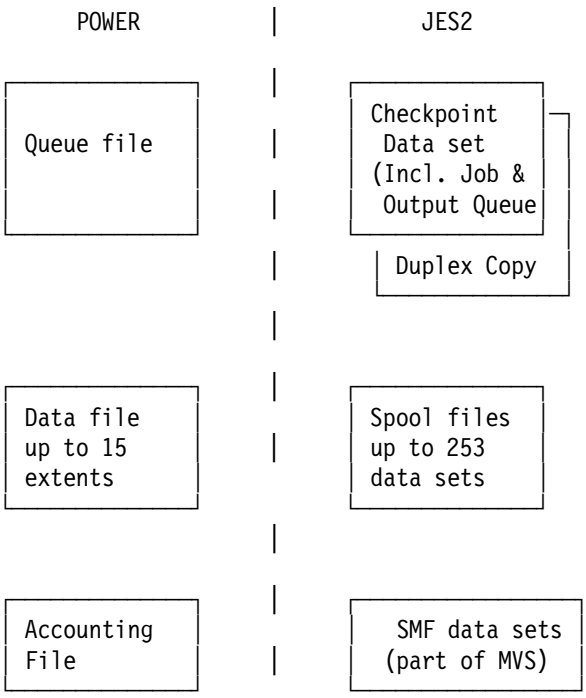
- Chapter 4, “Job Control Language (JCL) Differences and Considerations” on page 69
- Chapter 28, “Orientation to OS/390 Console Operation” on page 443

10.2 Implementing JES2

This section describes the significant migration and conversion activities to implement JES2 for the first-time user.

10.2.1 Setting Up the Required Resources

Both POWER and JES2 require DASD files to spool the jobs and their output, and queues to manage them. The following figure describes the various types of spool and control files of VSE/POWER and their equivalent file types in JES2.



Your starter OS/390 system has a small JES2 subsystem defined with a minimal JES2 checkpoint and spool data sets. You will have to redefine them to support your environment.

10.2.1.1 JES2 Checkpoint

Similar to the POWER Queue File (IJQFILE), JES2 uses the ″Checkpoint″ data set(s) to manage its queues. Two JES2 checkpoint data sets should be allocated on different volumes to contain a copy of the JES2 job and output queues and information that must be retained across a JES2 restart. For details on the size, placement, and specification, see Chapter 4 in the *OS/390 JES2 Initialization and Tuning Guide*.

10.2.1.2 JES2 Spool Volumes

One or more JES2 spool data sets must also be allocated to contain a copy of job input and output, along with spooled control blocks for JES2. All spool data sets must have the same data set name and reside on DASD volumes with serial numbers starting with the same four or five characters as specified on the SPOOLDEF parameters. All spool data sets must have the same names, so there can only be one per volume and the volumes can not be DFSMS managed. For the specific size, placement, and attribute specification, see Chapter 3 in the *OS/390 JES2 Initialization and Tuning Guide*.

10.2.2 Starting JES2

Similar to the startup options in VSE/POWER, you can start JES2 automatically at IPL time, or the operator can enter the ″S JES2″ command.

There are also two types of startup: cold start, and warm start. As with POWER, you must use the ″COLD″ option when bringing up JES2 for the very first time, or when restarting JES2 with incompatible init parms. Otherwise a warm start is required to preserve the jobs and spooled data from before. There are other

JES2 initialization options described in Chapter 1 of the *OS/390 JES2 Initialization and Tuning Guide*.

10.2.2.1 The JES2 Procedure

Similar to the POWSTRT procedure, the JES2 member of SYS1.PROCLIB is used to initialize JES2. (It must be in SYS1.PROCLIB, not in any other procedure library.) You must tailor the JES2 proc to include your JES2 load libraries, parameter members, procedure libraries, and other options. For the specific requirements of the JES2 procedure, see Chapter 1 in the *JES2 Init & Tuning Guide*.

10.2.3 Tailoring JES2

You can customize JES2 for your installation by setting JES2 initialization parameters, issuing JES2 operator commands, or using JES2 supplied exits.

10.2.3.1 JES2 Initialization Parameters

Similar to the VSE/POWER tables assembled with generation macros, JES2 uses a series of initialization parameters to tailor the system. Whereas the POWER macros are assembled and linked into the POWER phases, the JES2 initialization parameters are placed in one or more sequential or partitioned data sets, and pointed to by the JES2 procedure.

See Table 17 on page 226 for a comparison of POWER macros to JES2 initialization parameters.

10.2.3.2 JES2 Operator Commands

Practically everything you can specify in JES2 initialization parameters can also be specified or changed through JES2 operator commands.

See *JES2 Commands* for details.

See 10.4.3, “POWER-JES2 Command Equivalences” on page 231 for a comparison of POWER exits to JES2 exits.

10.2.3.3 JES2 Installation Exits

You can also customize JES2 through any of 49 supplied IBM exits, or modify the JES2 code directly through source modifications or VER/REP statements in the JES2 init deck.

See *JES2 Exits* and *JES2 Job Related Exits* for details.

See 10.4.2, “Exit Comparisons” on page 230 for a comparison of POWER exits to JES2 exits.

10.3 JES2-POWER Functional Comparison

Charts follow which compare differences between POWER and JES2 in the following areas:

- Input Services
- Job Scheduling
- Output Services
- Interface to VSE/ICCF or TSO
- Remote Job Entry (RJE)

- Network Job Entry (NJE)
- Application Programming Interfaces
- Accounting
- RAS Characteristics
- Testing Techniques

Note: The comparison is based on the functions provided within VSE/POWER. Therefore, it is not a complete overview of all functions available in JES2.

10.3.1.1 Multiple System Support

In POWER, the Spool File can be shared between multiple VSE/POWER systems using the “Shared Spool” feature. ***Multi-access spool*** or ***MAS*** is a standard feature of JES2. Even in a single system environment, JES2 assumes there are multiple systems (called ***members***) sharing spool and checkpoint.

JES2 allows a maximum of 32 members in an MAS, compared with POWER which allows only four. The JES2 members usually correspond to an MVS system, but you can have multiple JES2 subsystems (members) on the same MVS system.

Both VSE/POWER and OS/390 support the sharing of user files between multiple systems with integrity. This is supported by Global Resource Serialization (GRS) with OS/390. See *OS/390 V1R3.0 MVS Planning - Global Resource Serialization*, GC28-1759 for details.

10.3.2 Input Service

As with POWER, jobs may be submitted to JES2 from many sources. Here are some comparisons:

Table 11. JES2 Input Sources (compared to POWER)

Input From:	POWER	JES2	JES2 Comments
Local Card Reader	Y	Y	JES2 supports 2501, 2540, 3505 & 3525
Disk	Y (SLI)	Y	Use IEBGENER or IEBEDIT (RDRxx proc)
Tape	Y	Y	Use IEBGENER or IEBEDIT or RDRxx proc
3540 Diskette Reader	Y	N	3540 Reader Utility no longer supported
Remote BSC & SNA Terminal	Y	Y	See 10.3.6.2, “Remote Workstation Definitions” on page 219
Submit Function	Y (ICCF)	Y	TSO/E See 7.4, “Submitting Jobs for Batch Execution” on page 162
Internal Reader	Y	Y	// DD SYSOUT= (x,INTRDR)
NJE Nodes	Y	Y	See 10.3.7.1, “NJE Definitions” on page 221

10.3.3 Job Scheduling

POWER and JES2 both manage the job input queue and manage the job selection for execution according to job classes, priority and in the order they were submitted. In addition, OS/390 V2R4 provides additional capability with the workload management of batch jobs according to installation specified performance objectives.

Table 12. POWER/JES2 Job Scheduling Comparison

Job Scheduling Function	VSE/POWER	MVS/JES2	MVS/JES2 Comments
Job Selection Priority (Ranges)	Y (0 - 9)	Y (0 - 15)	
Job Classes	A-Z, 0-n n = number of partition	A-Z, 0-9	0-9 are treated just like letters
System Affinity	Y (via SYSID)	Y (SYSAFF)	Also see WLM Resource Affinity Scheduling Environments
Job Disposition (D/H/K/L)	Y	HOLD=YES	See 10.3.3.1, “Job Stream Disposition”
Schedule on a specific system	Y	Y (SYSAFF)	Also see Resource affinity scheduling with WLM in OS/390 V2R4
Time Event Scheduling	Y	N	Use a job scheduling package like OPC.

10.3.3.1 Job Stream Disposition

The VSE/POWER job dispositions are as follows:

D DELETE after processing.

H HOLD job. The job remains in the reader queue, it is not dispatched by VSE/POWER until the operator changes the disposition to D or K.

K KEEP after processing. The job will be automatically scheduled by VSE/POWER according to its class and priority. After job execution, the read queue entry is not deleted from the read queue, but the disposition becomes L.

L LEAVE in queue. The job remains in the read queue; it is not dispatched by VSE/POWER until the operator changes the disposition (LEAVE = HOLD + KEEP).

OS/390 Solution

Equivalent functions for disposition KEEP/LEAVE can be obtained via procedures causing an automatic resubmit, for example new read-in of the job as the last step using the internal reader function.

See 10.1.1.1, “KEEP Disposition for Pre-Execution Jobs” on page 207.

10.3.3.2 Serializing Job Execution

JES2 does not guarantee that jobs will run in the order they are submitted. If you need to make certain jobs run in order or you need dependent job control, you should submit them one at a time or use an automated job scheduling product such as OPC/A.

10.3.3.3 Time Event Scheduling

POWER supports the scheduling of job submission based on a one-time or repetitive schedule such as daily, weekly on a given day and time, and so on.

JES2 has a primitive *automatic command scheduling* facility, but you will probably need an automated scheduling package in OS/390 to do the same thing.

10.3.3.4 Additional Job Scheduling Functions with MVS/JES2

The following functions are not available in VSE/POWER, but may be exploited in OS/390 JES2:

Priority Aging When 'Priority aging' is on, JES2 will periodically increase a jobs priority for execution or printing, based on the length of time the job has been in the queue. For example, JES2 can be instructed to raise a job's priority by 1 every hour that it's been on the queue.

Time Limits A job's priority can be based upon its estimated elapsed time which can be specified on JECL statement or through JES2 exits. See the JOBPRTY JES2 initialization statement. (The installation can choose to cancel jobs exceeding their estimated time through initialization parameters or exits.)

The CPU time of a job can also be specified on the JOB or EXEC JCL statement, and controlled by SMF exits.

Output Limits The output priority can be based upon its actual output size according to the OUTPRTY JES2 initialization statement.

Job limits are based on JES2 estimated counts, JECL statements and JES2 exits, and can be extended or controlled by JES2 Exit 9.

Data set limits are based on the OUTLIM parameter of the DD statement. They can be extended or controlled by the IEFUSO SMF exit.

JCL Conversion This happens much earlier in the job's processing in OS/390 than in VSE. JECL (JES2 control cards) are processed at reader time. JCL is converted soon after reader time, then interpreted during step initialization.

Resource Affinity With OS/390 Version 2 Release 4, jobs can be selected according to the scheduling environment specified on the JOB card, and based on the availability or abstract resources as controlled by the workload manager.

Workload Managed Batch Initiators With OS/390 Rel.4, jobs can be selected based on installation defined performance objectives when running in "Goal" mode.

10.3.4 Output Service

As with POWER, JES2 supports line-mode printers, whereas PSF controls AFP Printers. (See Chapter 11, “Advanced Function Printing and Print Services Facility/MVS” on page 235 for AFP printing.)

Table 13 (Page 1 of 2). POWER/JES2 Output Service Comparison

Output Service Function	POWER	JES2	JES2 Comments
Local Line Printer (3211, 4245, 4248)	Y	Y	See 10.3.4.1, “Printers Supported” on page 216
Page Mode Printer (3900, 3820, etc.)	Y	Y	via PSF/MVS
Local Diskette	Y	N	(no longer supported in OS/390)
Remote BSC & SNA Printers & Punches	Y	Y	
Tape Spooling	Y	N	See 10.1.1.3, “Tape Spooling” on page 208
Interactive User	Y ICCF	Y	TSO/E Output or SDSF
Other NJE Node	Y	Y	
Output Classes	A - Z, 0 - 9	A - Z, 0 - 9	
Output Priority Ranges	0 - 9	0 - 15	
Output Disposition (D/H/K/L)	Y	Y	See 10.3.4.7, “Output Disposition” on page 217
Segmentation via JCL/JECL	Y (RBS Parm.)	Y	// DD SEGMENT= (line mode only)
Segmentation via Program Control	Y (SEGMENT macro or LFCB dynalloc)	Y	FREE=CLOSE or SPIN=UNALLOC dynalloc parm or SETPRT
Output Separation between Jobs	Y	Y	See 10.1.1.5, “Separator Page Difference” on page 208
User Info on Sep. Page	Y	Y	Use Exit 1 and/or 15 or JESNEWS
Forms Number	Y (1-4 chars)	Y (1-8 chars)	
Forms Control Buffer (FCB)	Y (1-8 chars)	Y (1-4 chars)	See 10.3.4.8, “FCB Naming Differences” on page 217
Universal Character Set (UCS)	Y (1-8 chars)	Y (1-4 chars)	See 10.3.4.9, “UCS Naming Conventions” on page 218
Multiple Destinations (RJE, NJE) via JCL	N	Y	
Printer Forms Alignment	Y (PSETUP cmd)	Y	See 10.1.1.4, “Printer Forms Alignment via PSETUP” on page 208

Table 13 (Page 2 of 2). POWER/JES2 Output Service Comparison			
Output Service Function	**POWER**	**JES2**	**JES2 Comments**
End-of-Page sensing	Y	N	See 10.1.1.6, “End-of-page Sensing” on page 209
Counting Line Mode Output	Y (Pages = skip to Ch.1)	Y (Lines)	Set PRINTDEF NEWPAGE=1
Counting Page Mode Output	Y (Pages)	Y (Pages)	
Restart capability for interrupted output	Y (PRESTART)	Y	Automatic checkpoint restart

10.3.4.1 Printers Supported

POWER and JES2 support the same set of channel-attached printers:

- 1403, 3211, 3203, 4245, and 4248 impact printers
- 3262-5 and 6262 as a 3211
- 3800-1 for line mode printing
- 3800-3, 3900, and other AFP printers through PSF

10.3.4.2 Output Segmentation

With POWER, the VSE installation has the ability to segment job output; this provides POWER the ability to print or punch output before a job is finished. It also allows the operator to have operational control over each segment created.

VSE/POWER supports output segmentation by record count thresholds, by specification in the input stream, and by specification in the program. Output segmentation can be used to improve turnaround time for jobs that have large volumes of output.

JES2 supports output segmentation with the SEGMENT= parameter on the SYSOUT DD statement (or on the dynamic allocation request). To support it at the installation level, use a JES2 Exit (6 or 31).

With JES2, job output processing (printing and punching) normally starts when the job completes. The MVS JCL parameter FREE=CLOSE or SPIN=UNALLOC on SYSOUT DD statements can be used to make these files available for printing or NJE transmission as soon as they are closed or unallocated. Then, the output produced for these SYSOUT files is processed when the files are CLOSEd by the application using them. Additionally, the application program can use MVS dynamic allocation services to spin off sections of output for immediate printing.

You can also use the SETPRT macro to schedule the data already written to spool for immediate printing.

10.3.4.3 Tape Spooling

Tape spooling is not supported in JES2. See 10.1.1.3, “Tape Spooling” on page 208 for some alternatives.

10.3.4.4 Printer Forms Alignment via PSETUP

The PSETUP function is not supported in JES2. See 10.1.1.4, “Printer Forms Alignment via PSETUP” on page 208 for some alternatives.

10.3.4.5 Separator Page Differences

The IBM provided separator pages are different with JES2. See 10.1.1.5, “Separator Page Difference” on page 208 for more information.

10.3.4.6 End-of-page Sensing

JES2 does not support end-of-page sensing. See 10.1.1.6, “End-of-page Sensing” on page 209 for more information.

10.3.4.7 Output Disposition

Conditional processing of output groups can be specified separately for normal and abnormal job termination. Use the // OUTPUT OUTDISP= statement with the same terms familiar to the VSE user (WRITE, HOLD, KEEP, LEAVE, PURGE). It can also be specified in JES2 initialization parameters on a job class or output class basis.

10.3.4.8 FCB Naming Differences

Both POWER and JES2 use eight-character FCB (Forms Control Buffer) names. Both also use four-character prefixes dependent on the printer device type. However, the prefixes are not similar, and the way they are specified is different.

FCB Prefixes

The four-character prefix of the FCB name is based on the device type and differs between POWER and OS/390.

Table 14. FCB Name Prefixes

Printer Device	VSE/POWER	OS/390
3203-1	FCB3	FCB2
3203-5	FCB2	FCB2
3211	FCB2	FCB2
3262-5	FCB2	FCB4
3800	FCB1	FCB3
4245	FCB2	FCB2
4248	FCB5	FCB4
5203	FCB4	FCB2
6262-14	FCB2	FCB4
PRT1	FCB2	FCB2
Device independent	$$$$	n/a
Note: The IBM 3262-5 and 4248 printers can also be run in 3211 compatibility mode, and use the FCB2 prefix.		

FCB Specification

POWER users specify the full eight-character FCB name, whereas OS/390 users only specify the last four characters. POWER supports device-independent specification of FCB-image phases in the * $$ LST statement by allowing ″$$$$″ as the first four characters. POWER then replaces the dollar signs by a character string depending on the printer:

```
FCB1  For a 3800 printer
FCB2  For a PRT1 printer (3211, 3203-5, 3289-4, 3262)
FCB3  For a 3203-1 printer
FCB4  For a 5203 printer
FCB5  For a 4248 printer
$$$$  For any other printer type
```

In OS/390, users specify the four-character name of the FCB and JES2 uses the four-character prefix based on the device type. The FCB and UCS images are stored in SYS1.IMAGELIB with device-dependent prefixes.

See *DFSMSdfp Advanced Services*, SC26-4921 and *DFSMS/MVS Utilities*, SC26-4926 for OS/390 details.

10.3.4.9 UCS Naming Conventions

In OS/390, Universal Character Set (UCS) images are stored in SYS1.IMAGELIB with device-dependent prefixes. POWER does not have any particular naming convention that must be followed.

JES2 supports four-character UCS names on JCL statements submitted by the user. Depending on the printer device type, the following ′UCSn′ name is prefixed to the name which is then retrieved from IMAGELIB:

- 1403 - UCS1xxxx
- 3203 - UCS2
- 3211 - UCS3
- 4245 - UCS5
- 4248 - UCS6
- 3262 - UCS6

The IBM 3800, when driven by JES2, uses character arrangement tables beginning with XTB1.

For details, see *DFSMS/MVS Utilities*, SC26-4926, and *DFSMSdfp Advanced Services*, SC26-4921

10.3.5 Interactive User Interfaces (ICCF/CMS/TSO)

Both VSE/POWER and OS/390 JES2 support interactive user interfaces for job submission and output retrieval, as well as other command functions. In addition, many VSE/POWER installations use VM/CMS as their interactive terminal system.

Table 15. POWER/ICCF, VM/CMS, and JES2/TSO Functional Comparison			
Interactive Interfaces	**POWER - ICCF**	**VM - CMS**	**JES2 - TSO/E**
Submit Jobs	Y	Y (Spool, Tag, Punch)	Y (TSO Submit)
Authorization Control	ICCF Administrator	-	RACF or Exits
Operating System Console Commands	Y	Y (SMSG)	Y (Extended Consoles, or SDSF)
Read Access to Spoolfile RDR/LIST/PUNCH	Y	Y	Y (TSO Output, or SDSF)
Inquire on Status of Jobs and Output	Y	Y	Y (TSO Status, or SDSF)
Cancel Jobs and Output	Y	Y	Y (TSO Cancel, or SDSF)

Jobs can be submitted with the TSO/E SUBMIT command, any ISPF EDIT panel, or SDSF using the SJ command on any job display.

Output can be browsed with the TSO/E OUTPUT command, the ISPF 3.8 panel, or SDSF using the O (output) or H (Held output) panels. The preferred interface is through SDSF.

10.3.6 Remote Job Entry

Both VSE/POWER and OS/390 JES2 support ″non-programmable″ Binary Synchronous Communication (BSC) remote workstations such as the IBM 2770, 2780, and 3780. Both support SNA single and multiple logical unit remote workstations such as the IBM 3770, and 3790.

Only POWER supports the IBM 3741 Data Station.

Only JES2 supports the multi-leaving BSC remote workstation such as the IBM S/360 Model 20 or 3773-2.

10.3.6.1 Functional RJE Differences

Both POWER and JES2 support remote and line passwords for remote workstation verification at sign-on time. Both support compression and SNA compaction.

10.3.6.2 Remote Workstation Definitions

Use the following JES2 initialization statements to define your RJE workstations and environment:

TPDEF Global definitions for RJE (and NJE) BSC & SNA buffer sizes, and number of SNA sessions supported simultaneously

LINE BSC and SNA TP Lines - these can be used for either RJE or NJE. See Table 18 on page 228 for a comparison of PLINE macros to JES2 LINE parameters.

RMT BSC and SNA RJE Workstations

R(nn).RD(n) RJE Workstation Readers

R(nn).PR(n) RJE Workstation Printers

R(nn).PU(n) RJE Workstation Punches

See Table 19 on page 228, and Table 20 on page 229 for a comparison of POWER PRMT macros and JES2 RMT parameters.

10.3.6.3 RJE Operations

See 28.6.1, “JES2 RJE Operations” on page 452 and *JES2 Commands* for a description of RJE commands for the host and remote operators.

10.3.6.4 RJE Exits

JES2 exits 17 or 18 can be used to control RJE remote sessions when they sign on. Exits 1 and 15 can be used for separators on remote printers. See Table 23 on page 231 for a comparison of POWER exits and JES2 exits.

10.3.7 Network Job Entry

Note

For a complete discussion of NJE differences, please see *NJE Installation, Operation and Use with JES2 and Other Systems*, GG22-9339.

NJE is supported by VSE/POWER, MVS/JES2, JES3, VM/RSCS, and some non-IBM program offerings. In general, all levels of VSE/POWER and JES2 are NJE-compatible with one another.

Both POWER and JES2 support SNA, BSC and CTC communications, although POWER only supports Virtual CTCs (using VM). Both systems support multiple streams, spanned headers, and AFP mode print files. JES2 also supports the following features which help manage NJE networks:

Path Manager This is exclusive to JES2 and dynamically keeps track of all connections, manages the best path and alternate paths to all nodes in the network.

Parallel Links You can have multiple parallel BSC lines, CTC connections and SNA sessions with adjacent NJE nodes for better availability and performance.

Multiple Paths Alternate path routing and multiple concurrent paths are supported by JES2 to adjacent and non-adjacent nodes.

Subnets You can define subsets of the NJE nodes as ″Subnets″ for routing purposes to simplify routing tables and path management.

Formatted Commands The JES2 $G commands allow you display and control jobs at other nodes in a system-independent format. (You don′t have to know the syntax of the target system.)

See Chapter 5 in the *JES2 Initialization and Tuning Guide* for more details.

10.3.7.1 NJE Definitions

Use the following JES2 initialization statements to define your NJE network and networking options:

TPDEF BSC & SNA Buffer Sizes, and Number of SNA Sessions

NJEDEF Number of Nodes, Transmitters, Receivers, and other NJE options

NODE Individual NJE node definitions

APPLID VTAM Applid of NJE nodes (if not the same as Node Name)

LOGON VTAM Applids of this node

LINE BSC, CTC, and SNA communication lines. These are defined in the same way RJE lines are defined. Lines can be used interchangeably between RJE and NJE. See the comparison of PLINE macro parameters and JES2 parameters in Table 18 on page 228.

CONNECT Predefined NJE Node Connections

See Table 21 on page 230 for a comparison of PNODE macros to JES2 NODE parameters.

In addition to the JES2 Init & Tuning Guide (Chapter 5), also see *NJE Installation, Operation and Use with JES2 and Other Systems*, GG22-9339.

10.3.7.2 NJE Operations

See 28.6.2, “NJE Operations” on page 453 and *JES2 Commands* for a description of NJE commands.

10.3.7.3 NJE Exits

JES2 exits 46 and 47 can be used to scan NJE headers when being sent or received, and the contents of the headers can be changed. JES2 does not have any exit to examine the spooled data records being sent with NJE. See Table 23 on page 231 for a comparison of POWER exits and JES2 exits.

10.3.7.4 NJE Management

See 10.3.7.1, “NJE Definitions” for detailed parameter comparisons.

10.3.8 Application Interfaces

10.3.8.1 Spool Space Allocation

POWER allocates spool space for jobs and spool files in units of DBLK groups which is usually about 4000 bytes. JES2 allocates space in units of track groups which are usually about three tracks each, as determined by the SPOOLDEF TGSIZE parameter.

10.3.8.2 Programmable Spool Interfaces

VSE/POWER supports macros to access spool data from user partitions: CTLSPOOL, GETSPOOL, PUTSPOOL, PWRSPL, MAPXPCCB, and XPCC.

There is not a one-for-one mapping of these macros, but the following APIs are useful for accessing JES2 spooled jobs.

Job Information Services

Current Job Identification

If you have code that is called by an MVS application, you can obtain information about the job currently running in an address space with the IAZXJSAB macro. You can retrieve information such as:

- Name of the subsystem that scheduled the job
- Job identifier
- Job name
- User ID associated with the job
- Time when the job started running
- JES status of the job

See *OS/390 MVS Auth Assm Services Reference ENF-ITT*, GC28-1765 for details.

Checkpoint Versions Subsystem Interface (SSI) function code 71 - the ″job information service″ provides an interface to acquire data from a copy of the checkpoint data. This allows you to find out about any job in the system known to JES2. See Appendix D in *JES2 Multi-Access Spool in a Sysplex Environment*, GG66-3263 for details.

Extended Status[6] Subsystem Interface (SSI) function code 80. This is an improved form of the STATUS SSI (function code 3). See *Using the Subsystem Interface*.

Output Retrieval

SYSOUT API[6] Use the SYSOUT Application Programming Interface (SSI function code 79) to retrieve output from JES2. This is an improved version of the PSO SSI (function code 1). See *Using the Subsystem Interface*.

Spool Data Set Browse Use this to dynamically allocate a spool data set and use standard I/O macros to read the file. See *OS/390 MVS Auth Assembler Services Guide*, GC28-1763.

Sample FSS[6] Use this as a model when writing your own Functional SubSystem, or as a tool for testing JES2 FSS printing. See *OS/390 Using the Functional Subsystem Interface*, SC28-1911.

Other Interfaces

Cancel Job You can cancel a job with SSI function code 2. See the ″SSCS″ mapping DSECT in the *OS/390 MVS Data Areas, Vol 5 (SSAG-XTLST)*, SY28-1168.

---

[6] Available in OS/390 Release 3.

Subsystem Version ID You can obtain version-specific information about a subsystem with SSI function code 54. See *Using the Subsystem Interface*.

Command Interface You can also use the MGCR service to submit JES2 or MVS operator commands to control jobs. See *OS/390 MVS Auth Assembler Services Guide*, GC28-1763.

10.3.9 Accounting Comparisons

10.3.9.1 Job Accounting

JES2 records the following accounting information:

- Job Purge - SMF Type 26 - This is the primary source of information about a job′s use of JES2 resources and history.
- Output - SMF Type 6 - Cut by JES2, XWTR, PSF and others
- NJE SYSOUT Transmit - SMF Type 57
- JES2 Subsystem Start/Stop - SMF Types 43, 45
- JES2 Spool Offload - SMF Type 24
- RJE/NJE Line Start/Stop - SMF Types 47, 48, 52, 53
- RJE Signon/Signoff - SMF Types 47, 48, 52, 53
- NJE Signon/Signoff - SMF Types 55, 58
- RJE/NJE Signon Password Violations - SMF Types 49, 56

JES2 SMF Accounting Records

SMF (System Management Facilities) is a function of OS/390 that collects and records various system and job data related to the use of resources. This information is recorded in the form of a number of different records, which are numbered. Installations can process the SMF records with any number of application programs to analyze the data, produce reports and so on.

JES2 uses SMF Type 26 (job purge) records to collect all successful SYSIN job transmissions, and SMF Type 57 records to collect all successful SYSOUT transmissions. Since multiple SYSOUT data sets may be transmitted within a job header and trailer, this record may represent multiple SYSOUT data sets.

Note that none of these record the node name of the local node. This can be a problem when combining records from multiple sites.

Job Purge Records Type 26 Records are cut when a job is purged from the system. The execution node name (and other execution-related fields) are not recorded in the type 26 record when the job executes on the origin node.

SYSOUT Transmission Records Type 57 Records are cut for each group of SYSOUT files sent to another NJE node, but do not contain the Jobname, or Time and Date on the reader at the original node.

NJE Network Management Records JES2 records the following information reflecting network events:

- SMF 55 Network Signon
- SMF 56 Network Integrity (invalid password)
- SMF 58 Network Signoff

See *System Management Facilities* for details.

NJE Accounting

Most NJE related information is carried in the NJE headers as the job is routed from node to node. Here is a summary of the differences between POWER and JES2 accounting records for NJE:

Table 16. Accounting Records for NJE Activities

NJE Activity	VSE/POWER Account Record	MVS/JES2 SMF Record
Job Transmission	Transmit/Receive Account Record	SMF26
SYSOUT Transmission	Transmit/Receive Account Record	SMF57
Job & SYSOUT Receipt	Transmit/Receive Account Record	- N/R -
Signons	- N/R -	SMF55
Signoffs	Network Account Record	SMF58
Start BSC Line	- N/R -	SMF47
Start SNA Line	- N/R -	SMF52
Stop BSC Line	- N/R -	SMF48
Stop SNA Line	- N/R -	SMF53
Line PW Violation	- N/R -	SMF49
Node PW Violation	- N/R -	SMF56

KEY: - N/R - Not Recorded

10.3.10 RAS Characteristics

JES2 has extensive recovery routines in the event of a failure within the spooling subsystem; many more than VSE/POWER. Here are some examples:

- Program check within the JES2 code
- Hardware Problem: Defective Track on Spool Disk Error Recovery and the JES2 “BADTRACK” initialization statement.
- Defective Spool Volume - Work continues without defective spool volume, replace if required
- Defective Track/Volume in Checkpoint - Checkpoint reconfiguration dialog
- Tape I/O Error - Error Recovery (not JES)
- Start JES2 without IPL with a Hot Start
- Spool Filling Warning Message

- Operator Monitor Spool Utilization
- Spool Full Condition - $S SPL upon warning (Output limits minimize this)
- An external message-based automation product can also delete, offload or reroute spool files.

10.3.11 JES2 Testing Techniques

Just as it is important to test new levels of OS/390 in your environment before using them in production, you should also test any JES2 exits or modifications before using them in production.

10.3.11.1 Poly-JES

Secondary JES2 subsystems, or “Poly-JES,” provide you with the ability to test new releases of JES2, and isolate JES2 work from the primary or production copy of JES2. Many installations find this convenient because it does not require a separate OS/390 test system.

There are two basic configurations for poly-JES:

1. Sharing the spool and checkpoint with the primary JES
2. Dedicating a unique spool and checkpoint to the secondary JES

See “Poly-JES” in Chapter 1 of the *JES2 Initialization and Tuning Guide* for detailed guidance. This section is only an addendum to that material. You must use a different character for the CONDEF CONCHAR parameter than the one used by the primary JES, or the secondary JES2 will not initialize.

The number of secondary JES2 subsystems that can be active is limited only by the number of different CONCHAR characters (22).

10.4 POWER/JES2 Detailed Comparisons

The remaining part of this chapter shows detailed comparisons between POWER and JES2 parameters and commands.

10.4.1 Mapping POWER Parameters to JES2 Init Parms

The following tables can help you transfer most of the parameters you have specified for POWER to their equivalent JES2 parameters.

10.4.1.1 Equivalent JES2 Parms for POWER Macro

In general, most of the JES2 initialization parameters have default values so you do not need to specify them. Many installations start with the sample deck in samplib - SHASSAMP - and modify those parameters as necessary.

You should review each of the parameters you specify that is different from the default value using the *JES2 Init & Tuning Reference* manual. Here is a short table to show you what POWER parameters can be specified in JES2:

Table 17 (Page 1 of 2). POWER Macro to JES2 Parameter Mapping

POWER Parm	Description	JES2 Parm	Comment or Recommendation
ACCOUNT	Job Accounting Information to be collected/recorded.	JOBCLASS TYPE6, TYPE26	SMF records are collected based on SYS(TYPE(nn:nn)) parm in PARMLIB(SMFPRMxx)
CLRPRT	Clear 3800 page buffer at EOJ	N/A	Option not supported by JES2.
COPYSEP	Produce separator pages (or cards) between data sets and copies.	PRT SEPDS	JES2 exit 15 required to write the separator pages (or cards).
DBLK	Size of data block transferred to tape and disk	SPOOLDEF BUFSIZE	Use the maximum (3992)
DBLKGP	Number of DBLKs allocated as a group.	SPOOLDEF TGSIZE	Default is 30
FEED	Eject and feed a new diskette at EOF	N/A	Not supported in JES2
JLOG	Display job-related messages on the console.	N/A	Use MPF exits to suppress specific JES2 messages
JOBEXIT	User-written exit for JCL & JECL	EXIT(2) EXIT(4)	For the JOB (2) other JCL & JECL (4) statements
JSEP	Extra separator pages or cards between job output	N/A	Use JES2 Exits 1 & 15 to insert extra pages or cards.
LTAB	Logical Carriage Control Tape (FCB)	N/A	Function not in JES2
MEMTYPE	Support SLI	N/A	Function not in JES2
MRKFRM	Use mark-form on 3800s between job output	PRT COPYMARK	(Only supported for FSS printers)
MPWD	Master password	N/A	
MULT12	Multiple channel 12 postings on FCBs	N/A	Function not in JES2
NETEXIT	User-written exit for input from NJE node	EXIT(47)	Scan the NJE headers (not for data records)
NTFYMSG	Maximum number of messages held for ICCF notification	N/A	No limits in JES2
OUTEXIT	User-written exit for output to printer or punch	EXIT(1) EXIT(15)	Job output group separator (1), or data set separator (15) (No exit is available for processing SYSOUT data records.)
PAUSE	Pause between each job output to a punch	PUN(nn) PAUSE	Use $S command to continue.
PNET	Include networking function (phasename)	N/A	Automatically included with JES2
PRI	Default job priority	JOBPRTY(n) PRIORITY	Job Priority based on estimated execution time. (See also RDR PRIOINC parameter and priority aging.)
RBS	Automatic output segmentation	N/A	Controlled by SEGMENT= on the application's DD Statement
RJEBSC	Support BSC RJE	N/A	JES2 always supports BSC RJE

Table 17 (Page 2 of 2). POWER Macro to JES2 Parameter Mapping

POWER Parm	Description	JES2 Parm	Comment or Recommendation
SECNODE	VSE Security ″Zone″	N/A	(Use RACF NODES class profiles for security.)
SHARED	Shared systems and accounting file	N/A	JES2 always assumes shared systems. Accounting files cannot be shared.
SNA= wscount, password, appl-id	Max number of SNA RJE stations VTAM ACB password VTAM APPLID	TPDEF RMTNUM LOGON P = LOGON A =	Max # of BSC and SNA Remotes Password Application ID for local JES2
SPLIM	Spool space utilization alert	SPOOLDEF TGSPACE= (WARN=	$HASP050 message issued at limit
SPOOL	Support XECB spool macros	N/A	
STDCARD	Punch card output limit	ESTPUN NUM=	Cancel job or allow based on OPT= and Exit 9
STDLINE	Print line output limit before warning message 1Q52I	ESTLINE NUM=	Use ESTPAGE NUM= for page-mode output
SUBLIB	For compatibility with previous releases of POWER	N/A	
SYSID	System ID for shared spooling	MASDEF OWNMEMB=	Let it default to SMFPRMxx SID( )
TIME	Active, idle, polling times for shared queues	MASDEF HOLD= DORMANCY=	Only use the defaults for single systems.
TRACESZ	Memory reserved for TP traces	TRACEDEF PAGES= TABLES=	Memory in extended storage (above 16MB)
XMTEXIT	User-written exit for output to an NJE node	EXIT(46)	Scan NJE Headers (not data)

10.4.1.2 PLINE Mapping to JES2 LINE Parameters for RJE and NJE

SNA lines are defined by specifying UNIT=SNA. BSC EP or CTC lines have several options:

Table 18. PLINE MACRO to JES2 Parameter Mapping			
PLINE Parameter	**Description**	**JES2 LINE Parameter**	**Comment**
ADDR=	Unit address of the BSC emulator port or CTC adapter	UNIT=	″SNA″ or unit address (4-digit addresses supported)
CODE	EBCDIC is the only value allowed. (ASCII is not supported)	LINECCHR=	EBCDIC or USASCII supported
INTRPT	Selector channel mode	N/A	Option not in JES2
MODSET	BSC (2701) Adaptor characteristics	INTERFACE= CODE=	BSC adapter interface & code
PSWRD	Line password	PASSWORD=	BSC or dedicated SNA line
SWITCH	Switched line or leased	N/A	Don′t specify LINE= on RMT statement for switched lines
TIMEOUT	Number of idle minutes before forced off	(not on LINE parm)	Specify DISCINTV= on the RMT(nnnn) statement.
TRNSP	Transparency feature	TRANSPAR=	Text transparency required for NJE

10.4.1.3 Define BSC Remotes

This table shows the conversion of POWER PRMT parameters to JES2 RMT remotes.

Table 19 (Page 1 of 2). PRMT MACRO to JES2 Parameter Mapping			
PRMT Parameter	**Description**	**JES2 RMT Parameter**	**Comment**
REMOTE	remote ID number	RMT(nnn)	′nnn′ is the remote ID number
TYPE	2770, 2780, 3741, or 3780 for BSC	TYPE	2770, 2780, or 3780/3781 for BSC (3741 not supported)
ABE	Additional buffer expansion on 2770 and 3741	BUFEXPAN=2	(3741 not supported)
BE	Buffer expansion on 2770	BUFEXPAN=1	
CS	Component selection	TYPE=3781 or 2770	
CSALST	Component select for LST output	N/A	Not specified at this level - based on device type (2770, 3781)
CSAMSG	Component selection for messages sent to remote	N/A	Use MSGPRT=Y to route messages to a printer
CSAPUN	Component select for punch output.	N/A	Not specified at this level - based on device type (2770, 3781)
HFC	Horizontal Tabs	HTABS	
LIST	Number of characters in print line	N/A	(not nec.)
LSTROUT	Default print routing	ROUTECDE	Same for LST and PUN
MRF	2780 multi-record feature	MRF2780	
MSG	width of message lines	N/A	(not nec.)
MSEEJCT	Suppress skip-to-1 after output before messages	N/A	

Table 19 (Page 2 of 2). PRMT MACRO to JES2 Parameter Mapping			
PRMT Parameter	**Description**	**JES2 RMT Parameter**	**Comment**
MSGSPCE	Suppress four space-3 commands after messages	N/A	
PUN	Width of card punch records	N/A	(always 80)
PUNROUT	default punch routing	ROUTECDE	Same for print and punch in JES2
REF	Short form for replication	N/A	Use ranges: RMT(m-n) ・
SCE	Space Compression/Expansion (2770, 3780)	COMPRESS	
TRNSP	Punch transparency	TRANSPAR	
TURNEOJ	Line turnaround required	(Based on device TYPE)	
Note: ・ JES2 does not support the ″Short Form″ of RJE definitions, but ranges may be used to define the same characteristics on many remotes: RMT(20-27) WAITIME=1,MFORM=J			

10.4.1.4 Define SNA Remote Workstations

This table shows the conversion of POWER PRMT parameters to JES2 RMT remotes.

Table 20. PRMT MACRO to JES2 Parameter Mapping			
PRMT Parameter	**Description**	**JES2 RMT Parameter**	**Comment**
REMOTE=	remote ID number	RMT(nnn)	′nnn′ is the remote ID number
LUT1 for SNA	TYPE=LUTYPE1 for SNA		
CMPACT	Compaction table name	COMPACT= YES|NO	Compaction table number specified on individual remote printer or punch
CONSOLE	Separate console device for messages	CONS	
LSTROUT	Default routing for LST output	Route	Applies to both PRT and PUN routing
LU	Logical Unit (LU) name	LUNAME	
MAXLRECL	Maximum logical record length	N/A	(not necessary)
PSWRD	Logon remote workstation password	Password	(use RACF)
PUNROUT	Default routing for PUN output	Route	Applies to both PRT and PUN routing
REF	Short form for replication	N/A	Use ranges: RMT(m-n) ・
SESSLIM	Maximum number of sessions (devices)	N/A	Based on number of devices specified in JES2.
XLATE	Translate Printed output from X′00′ - X′3F′ to X′40′	N/A	Specify TRANS on the R(nn).PR(m) statement.
Note: ・ See note in previous table.			

10.4.1.5 Define NJE Nodes

This table shows the conversion of POWER PNODE parameters to JES2 parameters.

Table 21. PNODE MACRO to JES2 Parameter Mapping

PNODE Parm	Description	JES2 Parm	Comment
NODE=	Name of the NJE node	NODE NAME=	
LOCAL	This is the local node.	NJEDEF OWNNODE=	
APPLID	VTAM Appl-ID	APPL(name) NODE=	Defaults to node name. (Use LOGONn for local node)
AUTH	Command authorization level	NODE AUTH=	
BUFSIZE	Transmission buffer size	TPDEF xxxBUF= (SIZE=	BELOWBUF for BSC or CTC EXTBUF for SNA
MAXBUF	Number of buffers for transmitters & Receivers	TPDEF xxxBUF= (LIMIT=	(shared between RJE & NJE)
PWD	Send or Receive signon password	NODE PASSWORD	Send=pwd for local node Verify=pwd for other nodes
ROUTE1	Indirect link if using store-and-forward (BSC, CTC only)	NODE SUBNET= -or- CONNECT	(use dynamic path management, or CONNECT statements and operator cmds)

10.4.1.6 Define Compaction Tables

Up to 99 different compaction tables can be defined in JES2. They can be used by SNA RJE or NJE. For RJE, these can be referenced by individual OUTPUT JCL statement, specified on an individual Remote or Remote Printer or Punch statement. For NJE, they can only be specified on a NODE or APPL basis. (Use these with caution; they may take more cycles than they are worth.)

Table 22. PCPTAB MACRO to JES2 Parameter Mapping

PCPTAB Parameter	Description	JES2 COMPACT Parameter	Comment
name	Name of the Compaction Table	NAME	Compaction tables can be referenced by name or number.
MASTER	3 to 16 master characters	CHAR= (nm,m1,m2, ...mn,	Number of master chars and n master chars
NOMASTn	Non-master characters	CHAR= (...nm1, ...nmx)	Remainder are non-master chars

10.4.2 Exit Comparisons

Here are the VSE/POWER exits and their equivalent exits in JES2. The major difference between the two subsystems, is that POWER allows you to scan and alter the source data, whereas JES2 only gives you access to the header information.

Table 23. POWER Exit to JES2 Exits			
POWER Exit	**Description**	**JES2 Exit**	**Comment**
JOBEXIT	Job input - Scan JCL & JECL (POWER exits allow access to SYSOUT data.)	EXIT(2) EXIT(3) EXIT(4)	JOB statement scan JOB accounting field Other JCL & JECL (No access to SYSIN data.)
NETEXIT	Input from NJE node	EXIT(47)	NJE Headers Received (No access to SYSIN or SYSOUT data.)
OUTEXIT	Output to printer or punch (POWER exits allow access to SYSOUT data.)	EXIT(1) EXIT(15)	Job Separator Data Set Separator (No access to SYSIN or SYSOUT data.)
XMTEXIT	Output to an NJE node	EXIT(46)	NJE Headers Transmitted (No access to SYSIN or SYSOUT data.)

10.4.2.1 Source Code Modifications

Most JES2 modules are distributed in source form and can be modified to meet specific customer needs, though this is not recommended by IBM. It is sometimes easier to modify the JES2 source code than accomplish the same thing through exits.

10.4.2.2 The JES2 Patching Facility

Patch and AMASPZAP statements can be used to make minor and temporary modifications to the JES2 object code until JES2 is restarted by directly replacing the changed code. The JES2 Patching Facility changes only the memory copy of data; the copy residing on DASD (for example, LPA) cannot be replaced.

For details and precautions, see the section entitled ″The JES2 Patching Facility″ in Chapter 1 of the *JES2 Initialization and Tuning Guide*.

10.4.3 POWER-JES2 Command Equivalences

There are major differences between JES2 and POWER. This section will compare the products from a different perspective, that of the central operator. Through the migration, the functional role of the operator will remain the same. Therefore the most often asked questions by the central operator will be the following,

1. How do I do a certain function?
2. In POWER, I used ′xyz′ command. What command do I use in JES2?

The following figure gives an overview of POWER commands and the JES2 equivalent. See *OS/390 JES2 Commands* and *VSE/POWER Administration and Operation* for details.

10.4.3.1 Queue Management Commands

Table 24. Queue Management Commands

POWER Command Code	PWR Short Form	Function	JES2 Command Verb
PALTER	A	Alter processing attributes of a POWER job or a POWER controlled partition	$T
PDELETE	L	Delete queue entries	$P
PDISPLAY	D	Display the status of jobs, messages, resources, and the network	$D
PHOLD	H	Put a job of a queue in hold/leave state	$H
POFFLOAD	O	Save or restore entries of a queue.	$S OFF
PRELEASE	R	Release a POWER job for further processing	$A

10.4.3.2 Task Management Commands

Table 25. Task Management Commands

POWER Command Code	PWR Short Form	Function	JES2 Command Verb
PACT		Activate a transmitter or a receiver.	$S
PCANCEL	C	Cancel a POWER status report or a job in execution.	$C
PDRAIN	N	Discontinue transmission or reception of jobs and or output by a given task.	$P
PEND		Terminate POWER option.	$P
PFLUSH	F	Terminate the work currently in process for a task and allow the task to continue with subsequent work	$C
PGO	G	Reactivate a task or partition.	$S
PRESTART	T	Restart a writer task.	$B, $N
PSTART	S	Place a partition under control of POWER, start a task, or initiate a session between two nodes.	$S
PSTOP	P	Release a partition from POWER, stop a task, or end a link or session between two nodes.	$P

10.4.3.3 Control Commands

Table 26. Control Commands

POWER Command Code	PWR Short Form	Function	JES2 Command Verb
PACCOUNT	J	Save account file records.	(SMF)
PBRDCST	B	Transmit a message.	$DM, $M
PINQUIRE	I	Display the status of a BSC line, SNA logical unit, or a node.	$D U,... $D Node
PLOAD		Load a network definition table.	$T Node
PRESET		Reset active jobs in a shared spooling environment.	$E
PSETUP	U	Print a page layout.	(see Note)
PXMIT	X	Route commands to another node.	$G $N

Note: There is no equivalent function in JES2. See 10.1.1.4, “Printer Forms Alignment via PSETUP” on page 208.

10.4.3.4 NJE Operator Commands

This section is a summary of NJE operator commands.

Network Management

The following tables provide a general reference of the operator commands available on the various systems. Because of fundamental differences between the systems, commands in the same row may not be identical. Refer to the specific product operations guides listed above for details.

Table 27 (Page 1 of 2). Network Management Commands

Function	POWER	JES2
Start Lines	N/A	$SLNEnnn
Start Networking (BSC)	S PNET,node,,line	$SN,N=node
Enable VTAM ACB	N/A	$SLGNn
Start Networking (SNA)	S PNET,node	$SN,N=node
Start Transmitters	PACT PNET,node,TRn	$SLn.JTm,Ln.STm
Start Receivers	PACT PNET,node,RVn	$SLn.JRm,Ln.SRm
Drain Transmitters	N PNET,node,TRn	$PLn.JTm,Ln.STm
Drain Receivers	N PNET,node,RVn	$PLn.JRm,Ln.SRm
Drain Lines	N/A	$PLNEnnn
Stop Lines Immediately	N/A	$ELNEnnn
Cancel Line Activity	F PNET,node	$CLNEnn
Reset Lines	N/A	$ELNEn,LNEm,...
Stop Networking	PPNET,node,EOJ	$PLNEn,LNEm,...
Stop Networking Immediately	PPNET,node	$ELNEn,LNEm,...

Table 27 (Page 2 of 2). Network Management Commands		
Function	**POWER**	**JES2**
Display Network Connections	I ALL	$DCONNECT, $DPATH
Display Nodal Attributes	I NODE = node	$DNODE
Alter Nodal Attributes	PLOAD PNET,ndt	$TNODE
Alter Route Tables	PLOAD PNET,ndt	$TNODE,SUBNET=, $TCONNECT
Define Nodes	DEFINE node	$ADD NODE
Trace Line Problems	S PNET,node,,,, TRACE	$TR,ON,ID= + 4 + 5

N/A = No available operator command

File Control

The following table shows the various operator commands that control jobs (SYSIN or SYSOUT, ″files″ to RSCS) on the various systems. Note that there are also global ($G..) commands for JES2 which can be used to control jobs on other nodes.

Table 28. File Control Commands		
Function	**POWER**	**JES2**
Display Network Queues	PDISPLAY XMT	$DQ,Q=XMT[node] $DQ,R=
Reorder network Queues	A XMT,job,PRI=	$TJnnn,P= $TOJnnn,P=
Display Job Routing	D RDR,job D LST,job	$DJnnn $TOJnnn
Re-Route Jobs	A XMT,job,NODE=	$RXEQ,D=
Re-Route Output	A XMT,job,NODE =	$RALL,D=node $TOJnn,D=
Hold Jobs and Output	H XMT[,job]	$HJnnn
Release Jobs and Output	CH splid NOHOLD	$AJnnn
Cancel Jobs and Output	C XMIT[,job]	$CJnnn, $PJnnn

Sending Commands and Messages

The following commands are available to system operators to send commands and messages to other nodes.

Table 29. Sending Commands and Messages		
Function	**POWER**	**JES2**
Send Message to Another Node	B node,′msg′	$DMNn,′msg′
Send Message to Interactive User	B node,user,′msg′	N/A
Send Command to Another System	X node,cmd	$Nnnn;cmd

Chapter 11. Advanced Function Printing and Print Services Facility/MVS

11.1 Introducing PSF/MVS

Print Services Facility (PSF) is the Advanced Function printing (AFP) printer-driver program for VSE and OS/390, as well as AIX, OS/2, VM and OS/400.

PSF has similar capabilities in all environments, plus differences unique to the operating system on which it is running. This chapter describes the differences between the VSE and OS/390 environments.

11.1.1 Functional Comparison between PSF/VSE and PSF/MVS

PSF/MVS supports all the printers and functions supported by PSF/VSE. In fact, you can regard PSF/MVS as a "super-set" of PSF/VSE. See *AFP: Printer Information*, G544-3290 for details.

11.1.1.1 Printers Supported

Both PSF/VSE and PSF/MVS support the same channel-attached printers, and SNA-attached network printers. In addition, PSF/MVS supports TCP/IP connected printers. (These are not supported by VSE/PSF.)

11.1.1.2 Printer Features

All features supported by PSF/VSE, such as "N-up" printing and ACIF (AFP Conversion and Indexing Facility) are also supported by PSF/MVS.

11.1.1.3 PSF/MVS Exclusives

The following features of PSF/MVS provide additional function:

- MVS Download to distribute output through PSF/6000 and AIX InfoPrint Manager
- NetSpool to capture output from VTAM print applications
- Installation Exits for separator pages, record transmission, and resource management
- Microfilm Printing[7]
- PSF-Direct printing to bypass JES spooling

11.1.2 Migration Effort

PSF/VSE and PSF/MVS have many common components and externals. Therefore, the effort to migrate is minimal and consists of the following major tasks:

[7] Microfilm printing is enabled for Anacomp film printing systems.

- 11.2, “Installing and Configuring PSF/MVS”
- 11.3, “Setting up AFP Resources” on page 240
- 11.3.4, “Migrating Print Applications” on page 241
- 11.4, “Understanding Operational Differences” on page 242
- 11.5, “Other Differences” on page 243

Other tasks are similar between the two platforms.

11.2 Installing and Configuring PSF/MVS

PSF/MVS is an optional feature of OS/390, and is already installed on your SystemPac or part of your ServerPac. If you are already licensed for Print Services Facility/MVS (PSF/MVS), 5695-040, you must explicitly enable it.

If you order the product with OS/390, the tailored IFAPRD00 member that IBM ships with your order contains the required PRODUCT statements. Otherwise, you must explicitly enable the licensed program when you run it with OS/390.

The following example shows the entry that you must include in the IFAPRDxx parmlib member to enable PSF/MVS:

```
PRODUCT OWNER('IBM CORP')
        NAME(PSF/MVS)
        ID(5695-040)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(PSF/MVS)
        STATE(ENABLED)
```

This should already be enabled with your ServerPac or SystemPac.

11.2.1 Defining Channel-attached Printers to MVS

This is similar to VSE. Use IOCP as described in Chapter 2 of the *PSF/MVS Systems Programmers Guide* to define parallel or ESCON channel-attached printers.

11.2.1.1 Attachment Options

PSF/MVS supports printers as system output devices for deferred printing through JES2, or for direct printing under Direct Printer Services Subsystem (DPSS). This way, the application program sends records directly to an attached printer (or directly to PSF), bypassing the JES spool. See Chapter 8 in the *PSF/MVS Systems Programmers Guide* for details.

11.2.2 Defining Network Printers

11.2.2.1 SNA-Attached Printers

This is similar to VSE. See Chapter 3 in the *PSF/MVS Systems Programmers Guide* for details.

11.2.2.2 TCP/IP Attached Printers

Unlike VSE, OS/390 also supports printing to TCP/IP connected printers.

There are several ways to print from PSF to LAN connected printers through TCP/IP:

- IP PrintWay
- MVS/Download and PSF/6000 and InfoPrint Manager
- MVS/LANRES
- TCP/IP for MVS Network Print Facility
- Directly attach Network IPDS printers with IPDS cards, IP 4000 systems with TCP/IP attachments

The IBM recommended method is through IP PrintWay, which is an optional feature of OS/390 or feature of PSF/MVS.

See the *IBM IP PrintWay Guide*, S544-5379.

11.2.3 The PSF Startup Procedures

PSF/MVS runs partially within the JES2 address space (when using deferred printing), and partly as a Functional Subsystem Application (FSA) in one or more separate MVS address spaces.

See Chapter 7 ″Using Deferred-Printing Mode″ in the *PSF/MVS Systems Programmers Guide* for sample initialization statements and startup procedures.

11.2.4 Defining Printers for PSF Printing

PSF driven printers, whether locally attached or network attached, must be defined by JES2 PRT(nnnn) statements in the JES2 initialization deck. They can also be added dynamically with the $ADD PRT(nnnn) command. See Chapter 7 “Using Deferred-Printing Mode” in the *PSF/MVS Systems Programming Guide* and the *JES2 Initialization and Tuning Reference* for detailed descriptions of the JES2 parameters.

Below is a simple example of the JES2 FSS and PRT statements from the *PSF/MVS Systems Programming Guide*:

```
FSS(FSS1) PROC=SAMPPROC,       /* Procedure name to start FSA   */
          HASPFSSM=HASPFSSM    /* Standard JES2 FSI Support Mod */

PRT1      FSS=FSS1,            /* Name of above FSS definition  */
          MODE=FSS,
          PRMODE=(LINE,PAGE),
          UNIT=20E,
          CLASS=A,
          UCS=0,
          SEP=YES,
          SEPDS=YES,
          CKPTPAGE=100,
          START=NO,
          MARK=YES,
          NPRO=99,
          TRKCELL=YES
```

11.2.5 FSS Procedure and PRINTDEV Statements

Below is the sample FSS proc shown in the *PSF/MVS Systems Programming Guide*.

```
//SAMPPROC PROC
//STEP01   EXEC PGM=APSPPIEP,REGION=4096K,TIME=1440
//STEPLIB  DD DSN=PSF.LINKLIB,DISP=SHR
//JOBHDR   OUTPUT PAGEDEF=V06483,       /* JOB HEADER PAGE           */
//         FORMDEF=A10120,CHARS=GT12    /*   FORMDEF: ALTERNATIVE BIN*/
//JOBTLR   OUTPUT PAGEDEF=V06483,       /* JOB TRAILER PAGE          */
//         FORMDEF=A10110,CHARS=GT12    /*   FORMDEF: MAIN BIN       */
//DSHDR    OUTPUT PAGEDEF=V06483,       /* DATA SET SEPARATOR        */
//         FORMDEF=A10110,CHARS=GT12    /*   FORMDEF: MAIN BIN       */
//MSGDS    OUTPUT PAGEDEF=A08682,       /* MESSAGE DATA SET          */
//         FORMDEF=A10110,CHARS=GT15    /*                           */
//FONT01 DD DSN=SYS1.FONTLIB,DISP=SHR   /* SYSTEM FONTS              */
//       DD DSN=INST.FONTLIB,DISP=SHR   /* INSTALLATION USER FONTS   */
//PSEG01 DD DSN=INST.PSEGLIB,DISP=SHR   /* INSTALLATION PAGE SEGMENTS*/
//       DD DSN=SPEC.PSEGLIB,DISP=SHR   /* SPECIAL PAGE SEGMENTS     */
//PSEG02 DD DSN=INST.PSEGLIB,DISP=SHR   /* INSTALLATION PAGE SEGMENTS*/
//OLAY01 DD DSN=INST.OVERLIB,DISP=SHR   /* INSTALLATION OVERLAYS     */
//PDEF01 DD DSN=SYS1.PDEFLIB,DISP=SHR   /* SYSTEM PAGE DEFINITIONS   */
//       DD DSN=INST.PDEFLIB,DISP=SHR   /* INSTALLATION PAGE DEFS    */
//FDEF01 DD DSN=SYS1.FDEFLIB,DISP=SHR   /* SYSTEM FORM DEFINITIONS   */
//       DD DSN=INST.FDEFLIB,DISP=SHR   /* INSTALLATION FORM DEFS    */
//PRT1     CNTL
//PRT1     PRINTDEV FONTDD=*.FONT01, /* FONT     LIBRARY DD          */
//          OVLYDD=*.OLAY01,         /* OVERLAY  LIBRARY DD          */
//          PSEGDD=*.PSEG01,         /* SEGMENT  LIBRARY DD          */
//          PDEFDD=*.PDEF01,         /* PAGEDEF  LIBRARY DD          */
//          FDEFDD=*.FDEF01,         /* FORMDEF  LIBRARY DD          */
//          JOBHDR=*.JOBHDR,         /* JOB HEADER SEPARATOR OUTPUT  */
//          JOBTRLR=*.JOBTLR,        /* JOB TRAILER SEPARATOR OUTPUT */
//          DSHDR=*.DSHDR,           /* DATA SET HEADER SEPARATOR    */
//          MESSAGE=*.MSGDS,         /* MESSAGE DATA SET OUTPUT      */
//          BUFNO=5,                 /* NUMBER OF WRITE DATA BUFFERS */
//          PAGEDEF=A08682,          /* DEVICE PAGEDEF DEFAULT       */
//          FORMDEF=A10110,          /* DEVICE FORMDEF DEFAULT       */
//          CHARS=GT12,              /* DEFAULT FONT                 */
//          PIMSG=(YES,16),          /* ACCUMULATE DATA SET MESSAGES */
//          DATACK=UNBLOCK,          /* UNBLOCK DATA CHECKS          */
//          TRACE=NO,                /* BUILD INTERNAL TRACE         */
//          SETUP=FORMS              /* SETUP MESSAGE                */
//PRT1     ENDCNTL
```

11.2.5.1 Comparison of PRINTDEV Statement Parameters

In PSF/VSE and PSF/MVS, the PRINTDEV statement is part of the PSF startup job which defines the AFP printing environment and default print attributes. (In OS/390, the PRINTDEV statement is actually a JCL statement with "//" in columns 1 and 2.) Many of these parameters can also be specified or overridden on the JES2 PRT(nnnnn) initialization statement or on the user's // OUTPUT statement.

Most PRINTDEV parameters are supported identically between PSF/VSE and PSF/MVS. Exceptions are listed in the table below.

Table 30. PRINTDEV Parameter Comparison

PSF/VSE PRINTDEF Parameter	OS/390 Equivalent Parameter	Description and Comment
ASA	(not necessary)	ASA control records do not need conversion
CKPTPAGE	JES2 parm: PRT(nnnn) CKPTPAGE	Number of pages to be printed before a checkpoint is taken.
FONTPR	PSF APSUX07 Exit (Resource management exit)	Whether font pruning will occur.
JSEPS	JES2 parm: PRT(nnnn) SEP=	Use PSF exits APSUX01, -02, -03 for additional controls
LOGDEST	JES2 parm: PRT(nnnn) Routecde=	One to four logical destination names used to select output from spool
MESSAGE =(pagedef, formdef)	PSF PRINTDEV statement: MESSAGE=*.name	Name points to an // OUTPUT statement for Page and Form definitions for Messages
MRKFRM	JES2 parm: PRT(nnnn) MARK=	Marking of the separator pages.
NOTIFY	(No equivalent)	Whether PSF notifies the operator of print errors
NPRO=nnn	JES2 parm: PRT(nnnn) NPRO=ssss	Number of seconds before a nonprocess runout
OUTFONTS	(no parameter)	Use PSF exit APSUX07 to activate outline fonts.
RESCBUFS	(not necessary)	PSF/MVS loads resources a buffer at a time.
RESOURCE	(no parameter)	Use PSF exit APSUX07 to keep or delete resources.
SEPPAGE	PSF PRINTDEV statement: JOBHDR=, and JOBTRLR=	Names of the page and form definition for separator pages are specified on an // OUTPUT statement in the PSF startup proc.
UNIT	JES2 parm: PRT(nnnn) UNIT=/ccua	Physical unit address for locally attached printer.

11.3 Setting up AFP Resources

PSF/MVS supports resources in the system libraries defined in the PRINTDEV statement, and dynamically on a per-job basis via the USERLIB JCL statement.

11.3.1 Migrating Resources from VSE to OS/390

11.3.1.1 Defining Resources

If you have the source definition files for these resources, you can use the same process to define them on OS/390.

The same utilities are used on OS/390 as on VSE to define resources: PPFA for PAGEDEFs and FORMDEFs, and OGL for Overlays. Page Segments (PSEGs) can be created by GDDM or other licensed programs. See *AFP Application Programming Interface: Programming Guide and Reference*, S544-3872 for more information.

DCF (Script) is also supported in both environments.

11.3.1.2 Without the Source

If you don't have the source definition files (and can't recreate them) then you will have to use a tool to migrate them (″Mass Migration″).

There are some tools available from the IBM Printing Systems on their FTP site to help you convert VSE resources to MVS:

VSERES REXX exec to convert multiple VSE LIBR PUNCH phases to AFP resources

RESPUNCH Assembly program to extract an AFP resource from VSE and create an input file and JCL for RESMAKE

RESMAKE Assembly program to create an AFP resource in the MVS system from RESPUNCH

VSEREBLK PC and VSE tools for editing/creating resources on workstations and transferring them to VSE

APTRCONV Convert MVS resource back to VSE (shipped as part of PSF/MVS)

All of the resources for a specific print job can be extracted into a sequential (RECFM=VBM) Resource Object file using ACIF in VSE. ACIF will also create a printable MO:DCA (LIST3820) file from linedata using a PAGEDEF, exactly as PSF would do.

The Resource Object can also be concatenated to the front of a print job for any PSF or converted to a PDS (or sequential files in VM) using the FLAT2PDS exec. See 11.6.5, “Tools” on page 244 for the Internet location.

11.3.2 Remote-Resident Resources

APTRMARK (VSE) and APSRMARK (MVS) differ in a subtle, but important way: Resources such as fonts that are shipped with PSF/VSE are ″marked″ with an ″object origin identifier triplet″ for VSE. Use the APSRMARK utility to mark resources that will be printer-resident.

If you move resources marked with the VSE program APTRMARK to an MVS system, PSF/MVS will not accept them for printer-residency. You must run the

PSF/MVS utility APSRMARK against these ported VSE resources in order for PSF/MVS to consider these resources ′marked′ for printer residency. (Printer resident fonts shipped with PSF/MVS are already ′marked′ with the APSRMARK program.)

11.3.3 Transferring Print Streams - VSE and OS/390 Coexistence

You can use NJE to transmit AFPDS (also known as LIST3820 or MO:DCA) files from VSE to OS/390 or vice versa. Print files created on VSE should print on PSF/MVS, and vice versa. See some of above for exchanging files between VSE and MVS.

Sequential print files can also be downloaded to a workstation from POWER using the IND$FILE protocol. MO:DCA (AFP) files should be transferred as binary files. Mixed-mode and line data should usually be transferred BINARY CRLF because record lengths are not appended. Note that POWER will change ANSI carriage control to machine.

PSF/VSE, PSF/MVS and PSF/VM include the AFP Toolkit, an API used to create AFP documents. A more advanced version is available for OS/2, AIX and MVS called ″AFP ToolBox.″

11.3.4 Migrating Print Applications

In general, the OS/390 Application Programmer has an easier job defining and using AFP resources than his VSE counterpart. See the *PSF/MVS: Application Programming Guide*, S544-3673 for details.

11.3.4.1 JCL and JECL Differences

All the AFP parameters on the VSE * $$ LST statement are fully supported on the OS/390 // DD or // OUTPUT statements.

11.3.4.2 Printing from TSO

The OUTDES (TSO/E) statement also has all the parameters available on the VSE APTZPARM macro.

11.3.4.3 Assembler Programming Interfaces

In OS/390, it is a lot easier to specify AFP attributes. Much of the VSE coding for print file attributes, while easily done in MVS, is also unnecessary. JCL SYSOUT and OUTPUT statements provide all the functions. Features such as FREE and SPIN may eliminate much of the need for special coding.

VSE Printer PARM Macro

In VSE, the APTZPARM macro is used to create library members for a specific set of attributes. These members′ names are referenced in the $$ LST statement. All parameters on the APTZPARM macro are available on the OUTDES macro in OS/390:

```
CHARS       CKPTPAG
DATACK      FORMDEF
FORMS       PAGEDEF
PIMSG       PRMODE
TRC
```

In OS/390, you can use the PSF writer procedure to provide the defaults.

OS/390 Dynamic Allocation and Output Descriptor Macros

Traditional SYSOUT attributes can be specified on the DYNALLOC macro. AFP attributes can be specified on the OUTDES macro.

11.3.4.4 High Level Language Programming Interfaces

COBOL Applications

Creating AFP in COBOL is essentially the same between MVS and VSE. VSE code will run in MVS unchanged. MVS coding will require changes to FDs and File-Control moving to VSE due to the lack of the ability to specify DCB attributes in VSE JCL. See the AFPREBLK program in vsereblk on the Web site for an example of COBOL that runs in both environments. See also above the AFP Toolkit API, which supports COBOL and PL/I.

PL/I

Subject to the same cautions as COBOL, PL/I can also be used for manipulating AFP output.

REXX

In OS/390, REXX does not have the I/O limitations it does in VSE. There are many REXX examples on the Web site.

11.4 Understanding Operational Differences

Similar to the POWER commands in VSE, JES2 commands are used to control PSF printing in OS/390.

11.4.1 Starting and Stopping PSF

There are no PSF/MVS-specific commands. PSF/MVS is automatically started and stopped as a result of JES2 $S PRT and $P PRT commands. Changes to the printer setup attributes and selection criteria are also controlled through the JES2 $TPTR commands.

11.4.2 Command Comparison

Note that VSE/POWER command objects (“devname”) are the names on the PRINTDEV statement in the PSF startup JOB, whereas MVS/JES2 command objects (“nnnn”) are printer numbers defined to JES2 via the PRT(nnnn) statements.

Table 31 (Page 1 of 2). VSE - OS/390 Command Comparison

VSE or POWER Command	JES2 Command	Comment
POWER Command		
PSTART devname [,classes] [,PARM=..]	$S PRTnnnn $T PRTnnnn,CL=q $T PRTnnnn,R=dest	Start the printer - Change SYSOUT Class selected - Change destinations selected

Table 31 (Page 2 of 2). VSE - OS/390 Command Comparison		
VSE or POWER Command	**JES2 Command**	**Comment**
PSTOP devname [,EOJ | RESTART | FORCE.]	 $P PRTnnnn $I PRTnnnn F FSSx,FORCE,PRTnnnn	Drain the printer - wait for current job to finish - interrupt current job, restart from ckpt - Only use if nothing else works
PXMIT devname [,CLASS=] [,LOGDEST=.]	$T PRTnnnn [,CLass=] [,Routecde=]	Change printer selection criteria - SYSOUT Classes - Route codes (Logical destinations)
PFLUSH devname [,HOLD]	$C PRTnnnn $E PRTnnnn	Cancel output on printer - Restart output on printer
PRESTART devname [,count]	$B PRTnnnn,pages $F PRTnnnn,pages	Backspace printer Forward-space printer
PGO devname	$S PRTnnnn	Restart printer after setup or interruption
VSE Partition Command		
TRACE devname	$T PRTnnnn,TRace=Y $S TRACE(nn)	Enable printer for tracing Activate tracing: nn = 11, 12, 14 & 15
TERM devname	$P PRTnnnn $C PRTnnnn	Drain printer and then Cancel output on printer
TRAP devname		(Use MVS SLIP)
IGNORE devname	$T PRTnnnn,TRace=N	Turn off tracing.
SET FORMDEF= PAGEDEF=	-	Use PSF Exit APSUX07
SET JSEPS=	$T PRTnnnn,SEP=	(must restart FSA after $T) or use PSF Exit APSUX01, -02, -03

11.5 Other Differences

11.5.1 Performance

The same factors affecting performance in VSE also apply in OS/390. The speed of the printer, complexity of the output stream, transmission speeds, VTAM and NCP parameters, host and printer resources all figure into the performance of the printing.

11.5.2 Installation Exits

PSF/VSE has no installation exits. PSF/MVS provides installation exits for your use in coding and installing modifications to PSF functions. For example, with these exits, you can:

- Create your own separator pages, or modify separator-page formats supplied by PSF
- Modify, add, or suppress output records
- Modify system management facilities (SMF) type 6 records
- Inspect, redirect, or suppress PSF messages
- Manage resources

See Chapter 17 ″Using Installation Exits″ in the *PSF/MVS Systems Programmers Guide*, especially the section “Do′s and Don′ts” at the front of that chapter.

11.5.3 Accounting

PSF/VSE uses the POWER ACCOUNT=AFP (or =ALL) parameter to capture accounting information about printing through PSF.

In OS/390, this data is recorded by SMF in the Type 6 records written by PSF.

11.6 References

11.6.1 PSF/VSE Publications

- *VSE/ESA Program and Workstation Guide*, SC33-6509 (moving VSE files)

11.6.2 PSF/MVS Publications

- *PSF/MVS: Messages and Codes*, S544-3675
- *PSF/MVS: Diagnosis Guide and Reference*, G544-5462
- *PSF/MVS: Application Programming Guide*, S544-3673
- *PSF/MVS: MVS Download Guide*, G544-5294
- *PSF/MVS: System Programming Guide*, S544-3672
- *AFP: Printer Information*, G544-3290

11.6.3 Redbooks

- *PSF/VSE Application Programming Guide*, S544-3666
- *AFP Printing in a Cross-System Environment*, GG24-3765

11.6.4 Other Sources

See the IBM Printing Systems Company Web Site at ***http://www.printers.ibm.com/***. The “Software Solutions” directory contains many documents including the following PSF product descriptions:

- *PSF/VSE 2.2 New Functions and Enhancements* by Dave Pilcher
- *PSF/MVS 2.2 New Functions and Enhancements* by Randy Deaver

11.6.5 Tools

11.6.5.1 PSF Supplied Utilities

APTRCONV can be used to transmit resources from MVS and VM to VSE. (Shipped with PSF/MVS and PSF/VM.)

11.6.5.2 DITTO

See Chapter 20, “DITTO” on page 381.

11.6.5.3 Other Utilities

See Chapter 29, “Orientation for Utilities” on page 455.

11.6.5.4 Internet Locations

The IBM Printing Systems Company Web Site at ***http://www.printers.ibm.com/*** contains a “Tools” directory along with product support, software solutions and so on.

Specifically, the following tools are available from the

http://www.printers.ibm.com/tools.html site or FTP from **ftp://ftp.software.ibm.com/printers/tools/vseres/**.

- **resmake.assemble**
  - MVS program to create an AFP resource from the RESPUNCH program.
- **respunch.assemble**
  - VSE program to send AFP resource to MVS system via NJE as a punch file
- **vseres.exec**
  - REXX EXEC to convert VSE LIBR PUNCH phases to AFP resources
- **vseres.txt**
  - file describing these tools
- **vseres.zip**
  - IP package of all the above files for downloading to your PC.

11.6.6 Services

There are many services available to help you migrate your AFP applications from VSE to OS/390. See the IBM Printing Company Services Web page at

http://www.printers.ibm.com/asg.html/.

Part 3. Converting VSE Languages to OS/390 Languages

Chapter 12. COBOL

12.1 Introduction

This chapter introduces IBM COBOL for OS/390 and VM (program number 5648-A25), which is the COBOL compiler available with OS/390. It also outlines the differences between the three COBOL compilers that are available on VSE and COBOL for OS/390 and VM.

Various strategies for converting your VSE COBOL applications to OS/390 are considered. These strategies depend on the COBOL compiler you use on your VSE system.

The three COBOL compilers available on VSE are:

DOS/VS COBOL	program number 5746-CB1
VS COBOL II	program number 5668-958
COBOL for VSE/ESA	program number 5686-068

In addition, methods of solving problems, which can arise during a VSE to OS/390 conversion of COBOL programs, are considered.

The information presented in this chapter is not sufficient by itself to carry out a successful conversion from COBOL under VSE to COBOL for OS/390 and VM. You should study carefully the publications referred to in Table 32 on page 252 for more information. This chapter is intended to draw attention to the more obvious problems that can arise in such a conversion.

For information on which COBOL runs on which host operating system, see Table 34 on page 351.

12.1.1 General Comments on COBOL for OS/390 and VM

COBOL for OS/390 and VM is the COBOL compiler for your OS/390 system. COBOL for OS/390 and VM enhances the COBOL object-oriented support for OS/390 that was introduced in IBM COBOL for MVS & VM. COBOL for OS/390 and VM is based on IBM COBOL for MVS & VM, and includes such features as COBOL ANSI 85 Standard Language Support, intrinsic functions, Year 2000 Support, interlanguage communications, and the mainframe interactive debug tool full-function offering.

COBOL for OS/390 and VM uses Language Environment as its run-time environment. Language Environment provides common services and language-specific routines in a single run-time environment for C, C++, COBOL, FORTRAN, and PL/I.

If you order the Full-Function feature of COBOL for OS/390 and VM, you receive the IBM Debug Tool along with the compiler. This debugging component provides both interactive and batch debugging capabilities on the host.

12.1.2 Comparison of IBM COBOL Compilers

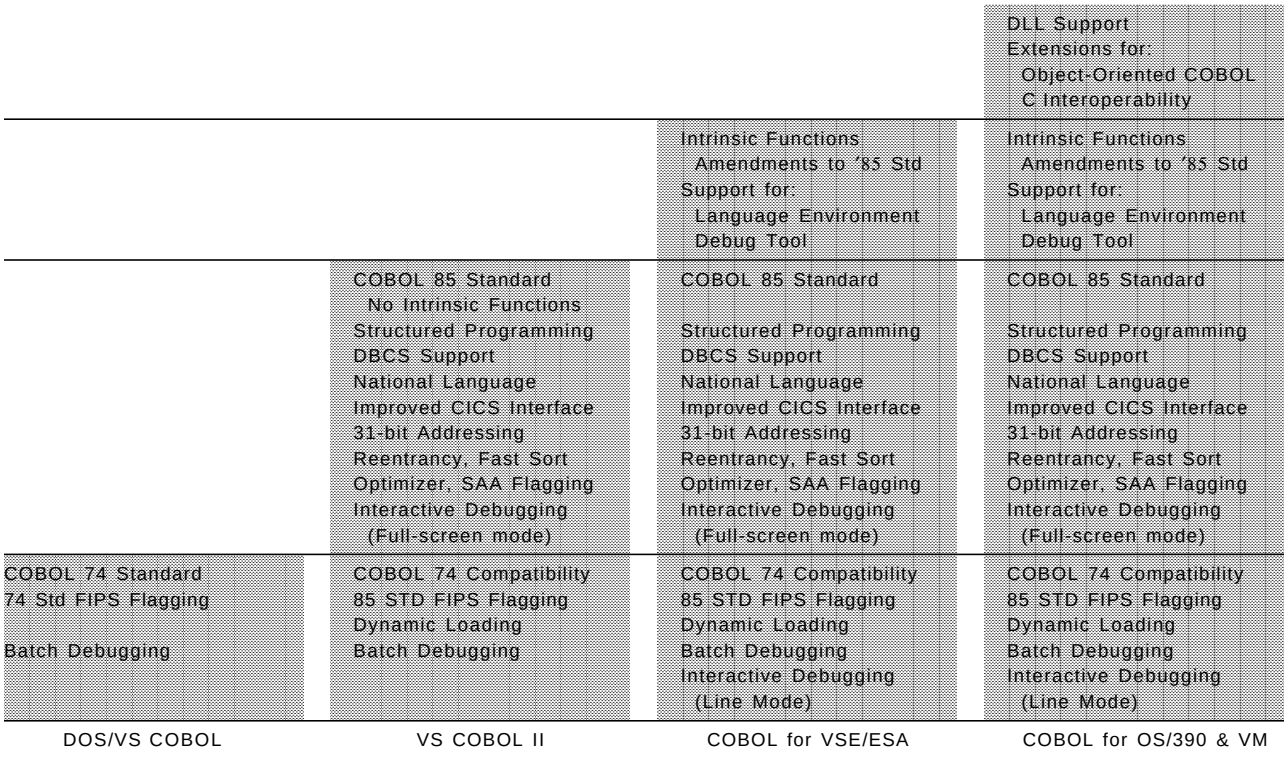


Figure 18. Comparison of IBM COBOLs

COBOL for VSE/ESA is source-compatible with COBOL for OS/390 and VM. Your COBOL for VSE/ESA programs should compile successfully without change under COBOL for OS/390 and VM.

VS COBOL II programs may require some changes to enable them to compile under COBOL for OS/390 and VM, but the changes will probably not be extensive.

DOS/VS COBOL programs will require modification before they will compile under COBOL for OS/390 and VM.

12.2 VSE to OS/390 Migration Considerations

The strategy you follow to migrate your COBOL applications to OS/390 depends on the COBOL compiler you are using in VSE, and the version of VSE you are running.

Up to and including VSE/ESA version 1 release 1, the only COBOL compiler available was DOS/VS COBOL. The conversion aid, COBOL and CICS Command Level Conversion Aid for VSE (CCCA/VSE) will not execute in this environment, but CCCA/MVS can be used.

Under VSE/ESA version 1 releases 2 and 3, the COBOL compilers available are DOS/VS COBOL and VS COBOL II; CCCA/VSE is available to aid the conversion process.

Under VSE/ESA version 1 release 4, and VSE/ESA version 2 and above, the COBOL compilers available were DOS/VS COBOL, VS COBOL II, and COBOL for VSE/ESA; CCCA/VSE is available to aid the conversion process. Now, only the COBOL for VSE/ESA product is available.

If you are running DOS/VS COBOL, you will have to convert your code to a new COBOL compiler level. Section 12.3, “Converting from DOS/VS COBOL” on page 252 outlines the various options open to you, to convert from DOS/VS COBOL.

Section 12.5, “Converting from VS COBOL II” on page 258 outlines some differences between VS COBOL II and COBOL for OS/390 and VM that you need to consider when migrating to COBOL for OS/390 and VM.

12.2.1 Migrating Object Code

If you are running VS COBOL II or COBOL for VSE/ESA, it is possible to transfer your compiled object code from VSE to your OS/390 system, linkedit with OS/390 Language Environment and run it there.

If you intend to do this, there are two compiler options to consider that affect the way your program runs under OS/390. These options are DECK and OUTDD.

DECK Use the DECK compiler option to produce an object program in a format that is suitable for migration to OS/390. The object program produced when the OBJECT compiler option is specified is not suitable for migration from VSE to OS/390.

OUTDD When you run a COBOL for VSE/ESA program under OS/390, the OUTDD compiler option is used to specify the name of the file for run-time DISPLAY output.

If you do not specify the OUTDD compiler option, the default is SYSOUT.

You should also check that there are no migration issues that you need to consider, between Language Environment for VSE/ESA Version 1 Release 4, and the release of Language Environment running in your OS/390 system. Language Environment for VSE/ESA 1.4, is functionally equivalent to Language Environment release 1.4 under OS/390. However, the release of Language Environment running in your OS/390 system will probably be much higher than 1.4.

Appendix C of the *COBOL for VSE/ESA Programming Guide* has more information on migrating COBOL for VSE/ESA object programs to OS/390.

12.2.2 Useful Publications

Table 32 on page 252 lists some publications that you may find useful when planning your conversion. Even if you are planning to convert directly to COBOL for OS/390 and VM, you may still find the *COBOL for VSE/ESA Migration Guide* useful. Also, *Taking Advantage of IBM Language Environment for VSE/ESA* has some useful tips on converting DOS/VS COBOL programs that also apply when converting to COBOL for OS/390 and VM.

Table 32. Useful COBOL Publications

Publication Title	Form Number
COBOL for OS/390 and VM Compiler and Run-Time Migration Guide	GC26-4764
COBOL for OS/390 and VM Language Reference	SC26-9046
COBOL for OS/390 and VM Programming Guide	SC26-9049
COBOL for VSE/ESA Migration Guide	GC26-8070
COBOL for VSE/ESA Programming Guide	SC26-8072
OS/390 Language Environment Migration Guide	SC28-1944
Taking Advantage of IBM Language Environment for VSE/ESA	SG24-4798

12.3 Converting from DOS/VS COBOL

If you are converting from VSE/SP or VSE/ESA 1.1, then you are running DOS/VS COBOL. In this case your source programs will have to be converted to COBOL for OS/390 and VM. Consider using a conversion aid such as COBOL and CICS Command Level Conversion Aid, running under OS/390, to help you.

If you are converting from VSE/ESA 1.2 or 1.3, then the conversion aid, CCCA/VSE, is available. CCCA/VSE converts your source code from DOS/VS COBOL to COBOL for VSE/ESA. COBOL for VSE/ESA source code is compatible with COBOL for OS/390 and VM, so you can then transfer your code to OS/390 for compilation and linkediting.

If you are converting from VSE/ESA 1.4 or VSE/ESA 2, then you also have the option to do a staged conversion. If you have COBOL for VSE/ESA and Language Environment for VSE/ESA installed on your system, you can convert your DOS/VS COBOL applications to COBOL for VSE/ESA and LE/VSE, and run them in your VSE system. When you are satisfied that they are running correctly, you can transfer the compiled object code to OS/390 for linkediting.

12.3.1 DOS/VS COBOL CICS Programs

OS/390 and COBOL for OS/390 and VM do not support CICS macro-level code. If you have any programs written in macro-level CICS you must convert them to command-level CICS.

COBOL for OS/390 and VM does not support BLL cells. If you use BLL cells in your CICS programs, you must modify the programs to remove them. CCCA will assist in making these changes.

When compiling DOS/VS COBOL CICS programs, the CICS translator did not require any particular option to indicate that the program being translated was DOS/VS COBOL. If you recompile your DOS/VS COBOL programs with COBOL for VSE/ESA, you must specify the CICS translator option `COBOL2`. If you recompile your DOS/VS COBOL programs under COBOL for OS/390 and VM, you must supply one of the CICS translator options `COBOL3` or `OOCOBOL`.

12.3.2 DOS/VS COBOL Programs Containing REPORT WRITER Statements

COBOL for OS/390 and VM does not support the REPORT WRITER statements. However, you can keep your REPORT WRITER statements by using the COBOL Report Writer Precompiler prior to the new compiler. Alternatively, you can use the COBOL Report Writer Precompiler to convert your REPORT WRITER statements to COBOL code.

12.4 DOS/VS COBOL and COBOL for OS/390 and VM Language Differences

Note

The following discussion on programming differences is relevant only to differences between DOS/VS COBOL and COBOL for OS/390 and VM.

12.4.1 Common COBOL Coding Problems

The following are some common DOS/VS COBOL coding 'mistakes' that will not work in OS/390. They may be logic errors or 'invalid' coding that DOS/VS COBOL nevertheless allowed. They will probably not be converted or notated by a conversion tool.

This is not meant to be an exhaustive list, but only some of the more common problems that can arise. You should read carefully the relevant chapters of either the *COBOL for VSE/ESA Migration Guide* or the *COBOL for OS/390 and VM Compiler and Run-Time Migration Guide* to determine the DOS/VS COBOL language elements that have changed or are no longer supported in COBOL for OS/390 and VM.

- Referencing a file's (or printer's) I/O area before the file (or printer) is OPENed will result in a system 0C4 abend in OS/390.
- Referencing a file's (or printer's) I/O area after the file (or printer) is CLOSEd will result in a system 0C4 abend in OS/390.
- A 'STOP RUN' statement should not be embedded within a SORT procedure. In OS/390, sorts must end before a STOP RUN can be requested.
- Level-88 statements that define non-numeric literals, when the literals are not enclosed in quotes, are invalid.

  For example:

  ```
  05 PRIMARY-FIELD PIC XX.
     88 FIELD1 VALUES ARE 60 61 62.
     88 FIELD2 VALUES ARE 50 51 52.
  ```

  Using COBOL for OS/390 and VM you will receive message:

  ```
  IGYGR1239-S Level-88 "VALUE" literal "61" was not compatible with the
  data category of the conditional variable. The literal was discarded.
  ```

  The literal values (60 61 62 50 51 52) must be enclosed in quotes. DOS/VS COBOL ignored the requirement for the quotes and processed the literals as the programmer intended.
- Redefinitions of level-01 entries in the File Section are not allowed. When more than one level-01 entry follows a file description entry, the redefinition is implicit.

For example:

```
01  RECORD-A                  PIC X(4).
01  FILLER REDEFINES RECORD-A.
    10  RECA-FIRST        PIC 9(2).
    10  RECA-SECND        PIC 9(2).
```

Using COBOL for OS/390 and VM you will receive the message:

```
IGYDS1064-E A "REDEFINES" clause was found in the definition of a
level-01 item in the "FILE SECTION". The clause was discarded.
```

This coding practice is documented as invalid in DOS/VS COBOL, but DOS/VS COBOL did not flag the error.

- With DOS/VS COBOL you can specify the SELECT OPTIONAL clause, for an input file that is to be accessed sequentially, and that may not be present each time the program is executed. However, if you do specify OPTIONAL, it is treated as a comment, since for DOS/VS COBOL this function is performed by the ASSGN job control statement with the IGN parameter.

  In COBOL for OS/390 and VM SELECT OPTIONAL is **required** for a file that may not be present each time the program is executed, and which is opened in input, I/O or extend mode.

  Therefore, if you have made use of the OPTIONAL key word only as a comment, you should remove it, as your program may produce unpredictable results.

- DOS/VS COBOL will accept the ACCEPT *identifier* FROM SYSIPT statement without the keyword FROM. COBOL for OS/390 and VM does not. It will generate the message:

  ```
  IGYPS2072-S "SYSIPT" was invalid. Skipped to the next verb, period or
  procedure-name definition.
  ```

- The program name supplied in the PROGRAM-ID paragraph is a user-defined word that identifies the program. If this name contains a ′ – ′ COBOL for OS/390 and VM will converted it to 0. This was true of DOS/VS COBOL also, but COBOL for OS/390 and VM generates a warning message, IGYDS0020-W, for example,

  ```
  IGYDS0020-W   Name "C2NAC-30" was processed as "C2NAC030".
  ```

- On returning to a COBOL calling program from an Assembler or other language subroutine, data is left in register 15. In DOS/VS COBOL it did not matter what this data was. In COBOL for OS/390 and VM the value in register 15 is passed to the RETURN-CODE special register. At the end of the program the value in the RETURN-CODE special register is returned to OS/390 as a user return code. If there was invalid data in register 15 on the return to the calling program, (and therefore also in the RETURN-CODE special register), the application may produce an unexpected return code from OS/390, or even a dump.

  This problem may be circumvented by adding the following statement to your converted source code:

  ```
  MOVE 0 TO RETURN-CODE
  ```

  You cannot make this change in advance of your conversion as the RETURN-CODE **special register does not exist in DOS/VS COBOL.**

12.4.2 ENVIRONMENT DIVISION

12.4.2.1 CONFIGURATION SECTION - SPECIAL-NAMES Paragraph

UPSI Switch Processing

In DOS/VS COBOL and COBOL for OS/390 and VM, program switches can be tested by use of a one-byte switch in the SPECIAL-NAMES paragraph, specified as UPSI-0 through UPSI-7.

In VSE, the setting of these program switches at execution time is achieved by the // UPSI job control statement.

In OS/390 the // UPSI job control statement is not available. Passing information at execution time is achieved by the PARM field of the EXEC statement.

You set your program switches at execution time by using PARM='/UPSI(nnnnnnnn)' on your OS/390 EXEC statement.

The following example shows an implementation of program switch processing using COBOL for OS/390 and VM in OS/390.

```
SPECIAL-NAMES.
    SYSIN  is ACCEPT-SYSIN
    UPSI-0 IS CBL232B ON STATUS IS CBL232-BASE
    UPSI-1 IS CBL232C ON STATUS IS CBL232-CURRENT.


//TEST  EXEC PGM=PROG1,PARM='/UPSI(10000000)'
```

12.4.2.2 INPUT-OUTPUT SECTION - I-O-CONTROL

MULTIPLE-FILE Clause (Tapes)

In DOS/VS COBOL this clause allows the specification of the relative positions of files on a multi-file unlabeled tape volume. In COBOL for OS/390 and VM it is syntax checked but has no effect on the execution of the program. The function is performed by the system through the LABEL parameter of the DD job control statement.

APPLY Clauses

In DOS/VS COBOL, there are seven formats for the APPLY clause:

```
APPLY WRITE-ONLY
APPLY EXTENDED-SEARCH
APPLY WRITE-VERIFY
APPLY CYL-OVERFLOW
APPLY MASTER-INDEX
APPLY CYL-INDEX
APPLY CORE-INDEX
```

In COBOL for OS/390 and VM only APPLY WRITE-ONLY is available. However the restrictions that applied to APPLY WRITE-ONLY in DOS/VS COBOL do not apply in COBOL for OS/390 and VM.

ASSIGN Clause

The format of the ASSIGN clause has become simpler. COBOL for OS/390 and VM may sometimes allow the DOS/VS COBOL format but may produce unexpected results at run-time. Refer to the *COBOL for OS/390 and VM Compiler and Run-Time Migration Guide* for more information.

12.4.3 DATA DIVISION - FILE DESCRIPTION (FD)

BLOCK CONTAINS

In OS/390 it is recommended that you specify BLOCK CONTAINS 0 RECORDS or BLOCK CONTAINS 0 CHARACTERS in your program. For an output file, you specify the required information on the DD statement. If you omit the blocking information for an output file OS/390 will supply a System Determined Blocksize (SDB). For an input file, the information is obtained from information in the catalog and VTOC.

If you specify BLOCK CONTAINS n RECORDS and also supply the information on the DD statement, for an output file, the BLOCK CONTAINS n RECORDS takes precedence over the DD statement information.

If you specify BLOCK CONTAINS n RECORDS for an input file, and the VTOC information does not match, your program may ABEND or return a file status code of 39.

LABEL RECORDS

DOS/VS COBOL accepts the LABEL RECORD IS *data-name* for non-sequential files. COBOL for OS/390 and VM does not, therefore you must change your program to remove LABEL RECORD IS *data-name* for these files.

LINAGE Clause and END-OF-PAGE Phrase

Under DOS/VS COBOL the END-OF-PAGE phrase may be specified without a corresponding LINAGE clause in the file description entry.

Under COBOL for OS/390 and VM if the END-OF-PAGE phrase is specified then the FD entry for the file must contain a LINAGE clause. Even then, you may find that your printed page layout is not as you expect. You should use your own line count logic in your program, making use of the LINAGE-COUNTER special register.

12.4.4 PROCEDURE DIVISION - Input/Output

CLOSE Statement for Tapes

Under DOS/VS COBOL the CLOSE *file-name* WITH LOCK statement closed and locked the file, and **UNLOADed** the tape reel or cartridge. Under COBOL for OS/390 and VM the file is closed and locked, but only rewound, not onloaded.

Similarly, for multi-volume tape files, DOS/VS COBOL rewinds and unloads each volume at end-of-volume. COBOL for OS/390 and VM only rewinds the tape, it does not unload it.

Note that this behavior may be different if you use a tape management system.

12.4.4.1 Program Termination

There are three COBOL program termination statements:

- EXIT PROGRAM
- GOBACK
- STOP RUN

There are some differences in the effect of these statements between DOS/VS COBOL and COBOL for OS/390 and VM. Table 33 gives a comparison of the behavior of these COBOL program termination statements, for DOS/VS COBOL and COBOL for OS/390 and VM.

Table 33. Action of COBOL Program Termination Statements

Statement		**DOS/VS COBOL**	**COBOL for OS/390 and VM**
EXIT PROGRAM	Main Program	No effect	No effect
	Subprogram	Return to calling program	Return to calling program
GOBACK	Main program	Abnormal job termination	Return to calling program (may be system and cause the application to end)
	Subprogram	Return to calling program	Return to calling program
STOP RUN	Main program	Return to system and cause end of job step (EOJ)	Return to calling program (may be system and cause the application to end)
	Subprogram	Return to system and cause end of job step (EOJ)	Return directly to calling program (may be system and cause the application to end)

12.4.5 File Handling Considerations

This section discusses some of the differences in file processing between DOS/VS COBOL and COBOL for OS/390 and VM.

12.4.5.1 File Status Codes

Many of the file status codes returned from file processing differ between DOS/VS COBOL and COBOL for OS/390 and VM. These differences and changes are summarized in the *COBOL for VSE/ESA Migration Guide* and the *COBOL for OS/390 and VM Compiler and Run-Time Migration Guide*. You should review the relevant sections of these publications carefully.

In particular, in the case of VSAM files, you will now need to refer to the VSAM feedback codes as well as the file status code, to determine the exact nature of the reported condition.

12.4.5.2 File Attribute Mismatches

DOS/VS COBOL file open processing does not always check that the attributes of the file definition in your program exactly match the file attributes of the physical file (for example, as defined for a VSAM file in the LISTCAT). To conform with ANSI 85 requirements, COBOL for OS/390 and VM open processing carries out many detailed checks for consistency between the program and actual file definition before opening the file. This can result in file open failures in COBOL for OS/390 and VM for files that were opened successfully in DOS/VS COBOL. You should add a file status check to your code following each OPEN statement. If these subsequently indicate problems you can amend your program accordingly.

12.4.5.3 ISAM

DOS/VS COBOL supports the processing of ISAM files, however COBOL for OS/390 and VM does not. Any ISAM files should be converted to VSAM Keyed Sequential Data Sets (VSAM/KSDS). CCCA/VSE can automatically convert the file definition and I/O statements from ISAM to VSAM/KSDS.

12.5 Converting from VS COBOL II

If your VS COBOL II source is VS COBOL II Release 4 and has been compiled with the NOCMPR2 option, then it is upward-compatible with COBOL for VSE/ESA, which, as we said earlier, is compatible with COBOL for OS/390 and VM. You can transfer your source code to your OS/390 system and recompile and linkedit it.

If you prefer, you can transfer the compiled object code to OS/390 for linkediting. See 12.2, “VSE to OS/390 Migration Considerations” on page 250.

However, if you are using LE/VSE callable services, you may need to change their names. Callable services which have names in LE/VSE beginning with CEE5.... are named CEE3.... in OS/390 Language Environment. These names will require changing and these programs will have to be recompiled in OS/390. You will not be able to transfer the compiled object code for these programs to OS/390. Refer to Chapter 17, “Language Environment (LE)” on page 351 for more information about migrating your run-time to Language Environment.

There are two new reserved words in COBOL for VSE/ESA, which you may need to consider in your conversion. They are:

- PROCEDURE-POINTER
- FUNCTION

If your VS COBOL II source is VS COBOL II Release 3.2, then, as well as the two reserved words mentioned above, there are three minor COBOL ANSI 85 Standard interpretation changes. These Standard interpretation changes affect the following language elements:

- REPLACE and Comment Lines
- Precedence of USE Procedures
- Reference Modification of a Variable-Length Group

12.5.1 VS COBOL II CICS Programs

COBOL for OS/390 and VM and OS/390 do not support CICS macro-level programs. If you have any programs written in macro-level CICS you must convert them to command-level CICS.

If you need to change your programs to cater for these differences, you can do so before you migrate them to OS/390. Then, if you prefer, you can transfer the compiled object code to OS/390 for linkediting. See 12.2, “VSE to OS/390 Migration Considerations” on page 250.

12.6 Converting from COBOL for VSE/ESA

If you are converting from COBOL for VSE/ESA then, with one exception, your source will be compatible with COBOL for OS/390 and VM. You can transfer your source code to your OS/390 system, recompile, and linkedit.

If you prefer, you can transfer the compiled object code to OS/390 for linkediting. See 12.2, “VSE to OS/390 Migration Considerations” on page 250.

The exception is that if you are using LE/VSE callable services you may need to change their names. Callable services which have names in LE/VSE beginning with CEE5.... are named CEE3.... in OS/390 Language Environment. These names will require changing and these programs will have to be recompiled in OS/390. You will not be able to transfer the compiled object code for these programs to OS/390. Refer to Chapter 17, “Language Environment (LE)” on page 351 for more information about migrating your run-time to Language Environment.

12.7 Some Conversion Considerations for all VSE COBOL Compilers

This section discusses some differences in the behavior of COBOL programs under VSE and COBOL for OS/390 and VM.

12.7.1 VSAM

Under VSE, if a VSAM file is not closed at the end of a program for any reason, (for example, no CLOSE statement in the program, or the program ABENDs), VSE will attempt an automatic close of the file. In this case, when the file is next OPENed, the file status code will be 0.

Under OS/390, if a file is not closed at the end of a program, the next time it is OPENed, OS/390 will perform an implicit VERIFY on the file, and successfully open the file. However, the file status code returned in this situation will not be 0, but 97.

Therefore, if your programs check for a successful OPEN, based on a file status code of 0, you should also check for a file status code of 97.

12.7.2 DISPLAY Statement

Under VSE, output from `DISPLAY` statements is interspersed with output from `WRITE` statements, and your programs may be coded to take account of this.

This does not happen in OS/390, as OS/390 has the ability to produce multiple print files and you should make use of this facility.

12.8 Compiler Options

This section discusses some of the compiler option considerations when converting from the various VSE COBOL compilers to COBOL for OS/390 and VM.

DOS/VS COBOL has many compiler options that are not available with COBOL for OS/390 and VM. Compiler options with VS COBOL II or COBOL for VSE/ESA are generally the same as COBOL for OS/390 and VM. If you are converting VS COBOL II programs, the most important difference to be aware of is the RES/NORES option.

12.8.1 RES/NORES

One compiler option is provided with VS COBOL II that is not available with either DOS/VS COBOL or COBOL for OS/390 and VM. This is RES/NORES.

Specifying RES in a VS COBOL II program causes the COBOL run-time subroutines to be loaded dynamically at run-time. NORES causes the run-time subroutines to be link-edited with the program.

DOS/VS COBOL behaves in a manner equivalent to specifying NORES with VS COBOL II. All run-time subroutines are link-edited with the program.

COBOL for OS/390 and VM behaves in a manner equivalent to specifying RES with VS COBOL II. The run-time is provided by Language Environment, and run-time subroutines are loaded dynamically at run-time.

In your conversion you should be aware of this different behavior.

12.8.1.1 DOS/VS COBOL Compiler Options not Available with COBOL for OS/390 and VM

Figure 19 on page 261 lists DOS/VS COBOL compiler options that are not available with COBOL for OS/390 and VM, and gives the COBOL for OS/390 and VM option you should use instead, if there is one. If you have used any of these options in your DOS/VS COBOL program, you should remove or change them.

DOS/VS COBOL Option	COBOL for OS/390 and VM Equivalent (If Any)
BUF=nnn	BUFSIZE(nnn)
CATALR/NOCATALR	NAME/NONAME
CLIST/NOCLIST	OFFSET/NOOFFSET
COUNT/NOCOUNT	None
FLAGE/FLAGW	FLAG(E)/FLAG(W)
FLOW/NOFLOW	None
LANGLVL(1/2)	None. COBOL for OS/390 and VM supports only the COBOL 85 Standard and COBOL 74 Standard (if using CMPR2 option) as implemented by VS COBOL II release 2.
LVL=A|B|C|D|NOLVL	None. COBOL ANSI 74 FIPS is not supported by COBOL for OS/390 and VM.
MIGR/NOMIGR	None. Not required by COBOL for OS/390 and VM.
PMAP=h	None. Obsolete option.
SPACEn	SPACE(n)
STATE/NOSTATE	TEST
SUPMAP/NOSUPMAP	COMPILE/NOCOMPILE
STXIT/NOSTXIT	None. Function not required.
SXREF/NOSXREF	XREF(SHORT)
SYMDMP/NOSYMDMP	ABEND dumps and dynamic dumps are available through Language Environment services. Symbolic dumps are available using the TEST compiler option.
SYNTAX/CSYNTAX/NOSYNTAX	COMPILE/NOCOMPILE
VERB/NOVERB	LIST/NOLIST
VERBREF/NOVERBREF	VBREF/NOVBREF
VERBSUM/NOVERBSUM	VBREF/NOVBREF
DECK/NODECK(LISTER)	None. The LISTER feature is not supported.
COPYPCH/NOCOPYPCH	None. The LISTER feature is not supported.
LSTONLY/NOLSTONLY	None. The LISTER feature is not supported.
PROC=1|2col	None. The LISTER feature is not supported.

Figure 19. Compiler Options Comparison DOS/VS COBOL and COBOL for OS/390 and VM

12.8.1.2 Compiler Option Considerations for VS COBOL II

The VS COBOL II and COBOL for OS/390 and VM compile-time environments are very similar. If you use the same compiler options that are specified in your current VS COBOL II applications, some internal changes may take effect, but basically the behavior is unchanged.

If you recompile your VS COBOL II applications with COBOL for OS/390 and VM and change compiler option settings, you should understand the possible effects on your applications. For information on compiler options with COBOL for OS/390 and VM see the *COBOL for OS/390 and VM Programming Guide*.

Figure 20 on page 262 lists the COBOL for OS/390 and VM compiler options that have special relevance to programs converted from VS COBOL II.

COBOL for OS/390 and VM Option	Comments
PGMNAME(COMPAT)	If compiling with COBOL for OS/390 and VM use this option to ensure that COBOL for OS/390 and VM processes program names in a similar manner as VS COBOL II.
RMODE(AUTO)	Use RMODE(AUTO) or RMODE(24) for COBOL for OS/390 and VM NORENT programs that pass data to programs running in AMODE(24).
TEST	The syntax of the TEST option is different in COBOL for OS/390 and VM than in VS COBOL II. The TEST option now has two suboptions; instead of specifying TEST, you now can specify a hook location and symbol-table location. TEST without any suboptions gives TEST(ALL,SYM). For more information on the TEST option, see *COBOL for OS/390 and VM Programming Guide*.
WORD(NOOO)	Use WORD(NOOO) if your existing programs use any of the object-oriented reserved words. For a list of these words, see Figure 26 on page 265.

Figure 20. Recommended COBOL for OS/390 and VM Compiler Options for Converted VS COBOL II Programs

Figure 21 on page 263 lists VS COBOL II compiler options that are not available in COBOL for OS/390 and VM. In some cases the function of the VS COBOL II option is mapped to a COBOL for OS/390 and VM option. This is described in the comments column.

VS COBOL II Option	Comments

FDUMP/NOFDUMP	COBOL for OS/390 and VM does not provide the FDUMP compiler option. For existing applications, FDUMP is mapped to the COBOL for OS/390 and VM compiler option TEST(SYM). This provides equivalent function and more.
	Language Environment generates a better formatted dump than VS COBOL II, regardless of the FDUMP option. But the presence of FDUMP enables Language Environment to include the symbolic dump of information about data items in the formatted dump.
	For information on how to obtain the Language Environment formatted dump at abnormal termination, see *Language Environment Debugging Guide and Run-Time Messages*.
	If FDUMP is used, COBOL for OS/390 and VM issues the warning message:
	`IGYOS4045-W The "FDUMP" option is not supported. This specification was interpreted as "TEST(NONE,SYM)".`
	If NOFDUMP is used, COBOL for OS/390 and VM issues the message:
	`IGYOS4003-E Invalid option "NOFDUMP" was found and discarded.`

FLAGSAA	COBOL for OS/390 and VM does not support the FLAGSAA option. If FLAGSAA is specified, COBOL for OS/390 and VM issues the message: `IGYOS4008-W The "FLAGSAA" compiler option was specified, but is not supported. The option was discarded.`

RES/NORES	COBOL for OS/390 and VM does not provide the RES/NORES compiler option. If RES is used, COBOL for OS/390 and VM issues the message:
	`IGYOS4046-I The "RESIDENT" option specification is no longer required. The resident runtime library support is always used.`
	If NORES is used, COBOL for OS/390 and VM issues the message:
	`IGYOS4047-W The "NORESIDENT" option is not supported. The resident runtime library support is always used.`

Figure 21. Compiler Options Comparison VS COBOL II and COBOL for OS/390 and VM

12.9 Reserved Words

This section discusses some of the reserved word considerations when converting from the various VSE COBOL compilers to COBOL for OS/390 and VM.

COBOL for OS/390 and VM has many reserved words that are not reserved with DOS/VS COBOL. There are two additional reserved words that are not reserved with VS COBOL II. There are additional reserved words in COBOL for OS/390 and VM for object-oriented COBOL that are not reserved in any VSE COBOL compiler.

12.9.1 Reserved Word Considerations for DOS/VS COBOL

COBOL for OS/390 and VM has reserved words that are not reserved in DOS/VS COBOL. They are listed in Figure 22 on page 264. If you used any of these words in your DOS/VS COBOL programs, you will need to replace them.

ALPHABET	END-COMPUTE	FALSE	OVERRIDE
ALPHABETIC-LOWER	END-DELETE	FUNCTION	PACKED-DECIMAL
ALPHABETIC-UPPER	END-DIVIDE	GLOBAL	PADDING
ALPHANUMERIC	END-EVALUATE	INHERITS	PROCEDURE-POINTER
ALPHANUMERIC-EDITED	END-IF	INITIALIZE	RECURSIVE
ANY	END-INVOKE	INVOKE	REFERENCE
BINARY	END-MULTIPLY	KANJI	REPLACE
CANCEL	END-PERFORM	LENGTH	REPOSITORY
CLASS	END-READ	LINAGE-COUNTER	RETURNING
CLASS-ID	END-RETURN	LOCAL-STORAGE	SELF
COMMON	END-REWRITE	METACLASS	SHIFT-IN
CONTENT	END-SEARCH	METHOD	SHIFT-OUT
CONTINUE	END-START	METHOD-ID	SORT-CONTROL
CONVERTING	END-STRING	NULL	SORT-MESSAGE
DAY-OF-WEEK	END-SUBTRACT	NULLS	STANDARD-2
DBCS	END-UNSTRING	NUMERIC-EDITED	SUM
DISPLAY-1	END-WRITE	OBJECT	SUPER
EGCS	EVALUATE	ORDER	TEST
END-ADD	EXTERNAL	OTHER	TRUE
END-CALL			

Figure 22. Reserved Words in COBOL for OS/390 and VM and not in DOS/VS COBOL

Some words which are not reserved in DOS/VS COBOL are COBOL ANSI 85 standard reserved words for a feature not supported by COBOL for OS/390 and VM. If used in a program, it is recognized as a reserved word and flagged with a severe message. These words are listed in Figure 23.

If you used any of these words in your DOS/VS COBOL programs, you will need to replace them.

CD	EMI	PRINTING	SUB-QUEUE-3
COMMUNICATION	ENABLE	PURGE	TABLE
DESTINATION	END-RECEIVE	QUEUE	TERMINAL
DISABLE	ESI	SUB-QUEUE-1	TEXT
EGI	MESSAGE	SUB-QUEUE-2	

Figure 23. Reserved Words in COBOL for OS/390 and VM for Unsupported Features

The words listed in Figure 24 are COBOL for OS/390 and VM compiler directing words. If they are used as a user-defined word, they will be flagged with a severe message.

If you used these words in your DOS/VS COBOL programs, you will need to replace them.

CBL	TITLE

Figure 24. Compiler Directing Words in COBOL for OS/390 and VM

12.9.2 Reserved Word Considerations for VS COBOL II and COBOL for VSE/ESA

There are two reserved words in COBOL for OS/390 and VM that are not reserved in VS COBOL II. These are shown in Figure 25. They are reserved in COBOL for VSE/ESA.

```
FUNCTION              PROCEDURE-POINTER
```

Figure 25. Reserved Words in COBOL for OS/390 and VM and not in VS COBOL II

The following words are reserved in COBOL for OS/390 and VM for the object-oriented COBOL extensions. They are not reserved in VS COBOL II or COBOL for VSE/ESA. Use the compiler option WORD(NOOO) if you are recompiling VS COBOL II or COBOL for VSE/ESA programs under COBOL for OS/390 and VM that use any of these words.

```
CLASS-ID              LOCAL-STORAGE  OBJECT           RETURNING
END-INVOKE            METACLASS      OVERRIDE         SELF
INHERITS              METHOD         RECURSIVE        SUPER
INVOKE                METHOD-ID      REPOSITORY
```

Figure 26. Reserved Words in COBOL for OS/390 and VM for Object-Oriented COBOL Extensions

12.10 Compiling and Running Your Converted COBOL Programs

Typically, job control procedures are used to compile, linkedit and run COBOL for OS/390 and VM programs under OS/390. Eight procedures are supplied with COBOL for OS/390 and VM to do this.

IGYWC	Compile only.
IGYWCG	Compile, load, and run. This is equivalent to the process of ′compile, link and go′ in VSE.
IGYWCL	Compile and linkedit.
IGYWCLG	Compile, linkedit, and run.
IGYWCPL	Compile, prelink, and linkedit.
IGYWCPLG	Compile, prelink, linkedit and run.
IGYWPL	Prelink and linkedit.
IGYWCPG	Compile, prelink, load, and run.

The use of these procedures is described fully in the *COBOL for OS/390 and VM Programming Guide*.

Chapter 13. Assembler

13.1 Assembler Products

In OS/390, the High-Level Assembler for MVS and VM Program Product (5696-234) is required for system generation (SYSGEN) and maintenance activities. It can also be used for application programming projects, and must be used when assembler routines are designed for 31-bit addressing facilities. See *High-Level Assembler for MVS and VM General Information*, GC26-4943 and *OS/390 MVS Extended Addressability Guide*, GC28-1769 for more information on this subject.

A Guide to Using MVS/XA Interface Facilities, SR21-1468 and SR21-1469, is recommended for installations that wish to **extend or customize system functions** provided by OS/390. The ″interface facilities″ described will use either exit or macro instructions to provide customization.

Recommendation

In converting assembler programs from VSE to MVS no attempt should be made initially to use 31-bit programming techniques. The main objective should be to get the programs ″converted to MVS programs as expediently as possible″; that is, don′t add new facilities (31-bit) at the outset. Once the programs are converted and operate successfully in MVS, they could then be reworked for 31-bit addressing if the need exists; for example, to address a VSCR problem.

13.2 General Assembler Conversion Comments

One of the most challenging tasks in moving from VSE to MVS may be the modification of application programs at the assembler language level. Coding at the assembler level includes control program macros and user-written machine language instructions. The machine language instructions are identical in both systems. The control program (or supervisor) macros and input/output macros of the systems are different, even though some have the same name. All base register usage, supervisor macros, and input/output logic of VSE assembler language programs must be checked for conformity to MVS conventions. A one-for-one mechanical replacement is possible in many cases. The complexity of the program and its use of supervisor functions is proportionate to the effort required to convert the programs. Simple programs are usually easy to convert.

Registers are an important factor in performance. When a VSE program is to be used under MVS, there may be mandatory changes in the use of registers because of macro expansions. One way to handle this problem is to add additional instructions to shift the MVS register contents to make them correspond to the VSE register contents. However, this approach may cause MVS to run slower because of unnecessary loading and storing of registers. The alternative is to change the register usage within the program to conform to MVS requirements, using the symbolic register notation through equates. Correct register usage in complex programs prevents problems requiring extensive programmer debugging effort.

The selection of a particular option of MVS may require redesigning the application programs. In addition, a program logic change may also be forced by attempting to simulate a VSE function under MVS. Examples of these possibilities include multitasking, interrupt handling, and communication region accessing.

The input and output components of the linkage editors, the job control language, and linkage edit control statements necessary to build program structures are discussed in the publication *DFSMS/MVS Program Management*, SC26-4916.

VSE assembler language programs that are changed to MVS must add an initialization routine to meet MVS requirements. You should establish a standard for the entire installation that can be simply inserted into the assembler language source member before it is recompiled under MVS. For additional information on the MVS services and macro coding details, refer to the following publication:

- *OS/390 MVS Programming: Assembler Services Reference*, GC28-1910.

For Data Management programming macro information, refer to:

- *DFSMS/MVS Using Data Sets*, SC26-4922.
- *DFSMS/MVS Macro Instructions For Data Sets*, SC26-4913.

Important

The macro functions and parameters described in this text may not be totally up-to-date. Macro facilities change over time. New macros and macro parameters become available with new releases of products. Therefore, you should always reference the appropriate macro manuals (listed above) for exact macro functions and applicable parameters. (Even though some macros may not be up-to-date, the techniques illustrated here should still be applicable.)

The next section highlights the services provided by the MVS supervisor and relates them to comparable ones provided by VSE. Information on VSAM macros is found in the section 13.2.5, “VSAM Macros” on page 290. Data Management macro comparisons are addressed in the section 13.2.6, “Data Management Macros” on page 292.

13.2.1 System Interface and Macros

The functions of the VSE system interfaces and macros and their MVS equivalents are discussed in the following text.

MVS Register Conventions

Application program use of general purpose registers in MVS is restricted to registers 2 through 12. (Registers 0, 1, 13, 14 and 15 are used for special purposes by MVS - see next sections.) If VSE programs use other than registers 2-12 for application purposes, program register assignments may have to be changed.

13.2.1.1 Initiation

Under VSE, main programs (those programs that are invoked by the operating system directly) are not required to save any registers upon entry. VSE assembly programs are not required to provide a save area unless that program invokes (calls) another program. In MVS, all programs are executed as subroutines including the program that is given control by the operating system. Therefore, all programs that are changed to MVS must ensure the presence of an initialization routine meeting MVS requirements. This initialization routine must do the following:

- Store all registers, except register 13, in the system or calling program save area (STM 14,12,12(13))
- Establish a base register - it must not be register 13 - to point to the start of this program.
- Provide a new save area in this program.
- Store the address of the calling (may be MVS) program′s save area in the called program′s save area (R13 + 4 - backward chain)
- Set up register 13 to point to this program′s save area if this program is to call subsequent subroutines or issue any system macros.
- Store register 13 in the calling program′s save area + 8 (forward chain)

This routine can be standard code established for the entire installation; it can then easily be inserted in the front of each program.

Note: Register 13 should not be used for any function other than pointing to a save area. For more information, refer to Figure 27 on page 270 and Figure 28 on page 271 for examples of this routine.

13.2.1.2 Termination

At job termination, VSE uses the EOJ macro which generates a supervisor call. The VSE supervisor, which maintains control of its own registers, then branches to the appropriate routine. Because this macro facility is not available under MVS you must return to the control program at the end of job as follows:

1. Restore the control program′s registers to their status upon entering this routine.
2. Branch to the return address stored in register 14.

You may also accomplish this function by using the MVS ″RETURN″ macro. The RETURN macro requires that Register 13 contains the address of the save area in the program that you are returning to.

Register Conventions

MVS linkage register conventions, upon entry to a routine, are compatible to those of VSE:

Reg. 13	Points to the calling program′s save area.
Reg. 14	Points to the return address in the calling program.
Reg. 15	Points to the entry point of this called program.
Reg. 0 and 1	Points to parameters or lists of parameters passed from the calling program to this called program.

```
PROGA START            PROGB CSECT            PROGC CSECT
      BALR                    (VSE)                  (VSE)
      USING                     .                      .
        .                       .                      .
        .                    ST 13,SAVEB+4          ST 13,SAVEC+4
        .                       .                      .
      Application            Application            Application
      Program                Program                Program
      Logic                  Logic                  Logic
        .                       .                      .
      LA 13,SAVEA            LA 13,SAVEB              .
      CALL PROGB             CALL PROGC               .
      EOJ(Return)            Return(PROGA)          Return(PROGB)
  SAVEA DC 18F'0'        SAVEB DC 18F'0'        SAVEC DC 18F'0'
        END                    END                    END
```

Figure 27. VSE Subroutine Linkage

These registers have additional meaning under MVS:

Register 1
: The PARM field of the EXEC statement provides an external facility for providing information to the program at job step execution time. When control is passed to your program, register 1 contains the address of a fullword on a fullword boundary. This fullword is the starting address of a 102 byte area on a halfword boundary where 100 bytes of information can be entered in the PARM field. This is one approach you can use to replace the VSE user communication region routines; for example, setting the UPSI byte externally.

Register 15
: Before returning to the calling program, you can load register 15 with a return code. You can analyze this code by MVS job control statements of a subsequent step to determine if the step is to be executed or bypassed. You can substitute return code processing for VSE logic which tests indicators set in the communication region by a previous step of the job.

Register 13
: Must point to a save area if any I/O or calls to subordinate programs are performed. If, in the VSE version of the program, register 13 is used as a base register, consider one of the following alternatives:

- If any unused registers are available, one of these should be substituted as a base register or,
- The save area may be placed in front of the program code that is to be based on register 13. Register 13 is then both pointing to a save area and acting as the base register.

Save Areas

Under both operating systems, all programs must provide a save area before calling another program. By convention, this save area should be 18 fullwords and contain the general purpose registers, as well as save area chaining pointers. The called program performs the storing of registers into the calling program's save area. The first program to receive control from MVS is responsible for saving the registers in the system-provided save area (address passed in register 13).

In MVS, the called program should provide a save area regardless of whether it calls additional subroutines. MVS Data Management functions, in certain instances, store the calling routine's registers into a save area whose address is

contained in register 13. Therefore, you must specify a save area to receive the registers.

```
PROGA START             PROGB CSECT             PROGC CSECT
      (MVS)                   (MVS)                   (MVS)
       .                       .                       .
       .                       .                       .
      STM 14,12,12(13)        STM 14,12,12(13)        STM 14,12,12(13)
      ST  13,SAVEA+4          ST  13,SAVEB+4          ST  13,SAVEC+4
      LA  11,SAVEA            LA  11,SAVEB            LA  11,SAVEC
      ST  11,8(13)            ST  11,8(13)            ST  11,8(13)
      LR  13,11               LR  13,11               LR  13,11
       .                       .                       .
       .                       .                       .
   Application             Application             Application
   Program                 Program                 Program
   Logic                   Logic                   Logic
       .                       .                       .
       .                       .                       .
      CALL PROGB              CALL PROGC
      L   13,4(,13)           L   13,4(,13)           L   13,4(,13)
      Return (MVS)            Return (PROGA)          Return (PROGB)
SAVEA DC  18F'0'        SAVEB DC  18F'0'        SAVEC DC  18F'0'
       .                       .                       .
      END                     END                     END
```

Figure 28. MVS Subroutine Linkage

If a standard save area of 18 fullwords is reserved in the calling program, the save area contains the following information at completion of the called program's initialization logic.

Word 1	Used by LE-compliant languages
Word 2	Address of the caller's save area (the backward chain).
Word 3	Address of the save area of the called program (the forward chain).
Word 4	Register 14. Return address within the calling module.
Word 5	Register 15. Entry point address of called module.
Words 6-18	Registers 0 through 12, respectively, of the calling program.

Consider three programs using the concept of forward and backward chains with standard linkage conventions. Under VSE, these could be three application programs, while under MVS, the highest-level program that must be considered is the MVS control program because it calls the MVS highest-level application program.

Linkage Macros

CALL, SAVE, and RETURN macros are available under VSE and MVS. This set of macros performs the general housekeeping required to maintain subroutine conventions within the CSECTs of a simple program structure. In general, these MVS macros provide additional functions not available in VSE. You can use the VSE versions of these macros under MVS without any modification.

```
VSE CALL    Entrypoint ,(PARAMETER LIST)
            (15)

MVS CALL    Entrypoint ,(ADDRESS),VL
            (15)       ,ID=number
```

Call is used the same way in MVS as it is in VSE, except when it is used with the LOAD macro. For a discussion of this difference, see the topic “LOAD Macro” on page 277 in this section. In addition, if a variable number of parameters may be passed, the VL keyword operand must be added. The parameters of the called module should be checked for VSE and MVS differences. If differences are found, make the necessary changes. See the following example.

```
     VSE                        MVS

   CALL SUBRTN1               CALL SUBRTN1
   CALL SUBRIN2,(TAB,BK)      CALL SUBRTN2,(TAB,BK),VL

VSE SAVE    (r1  ,r2)
MVS SAVE    (regl  ,reg2)   ,T  ,identifier name
```

Under MVS, the SAVE macro causes the contents of the specified registers to be stored in the save area at the address contained in register 13 (within the calling program). Use the SAVE, macro only at the entry point of a program. Do not use the SAVE macro in a program interruption exit routine. When you use the T and identifier name parameters, the code resulting from the macro expansion requires that register 15 contain the address of the SAVE macro.

The T operand specifies that registers 14 and 15 are to be stored in words 3 and 5 of the save area. It thus permits you to save two noncontiguous sets of registers. The identifier name operand is an identifier that aids in locating a program′s save area in a dump. It can be a complex name of up to 70 characters. Coding an identifier name causes the SAVE macro expansion to include:

- A count byte containing the number of characters in the identifier name. This byte is assembled four bytes following the address contained in register 15.
- The character string containing the identifier name. This string is assembled following the count byte.
- An instruction to branch around the count and identifier name fields.

Sample MVS SAVE:

```
 LAREX    CSECT
          USING    *,11           Establish Addressability
          LA       15,SAVEIT      Address of SAVE macro
 SAVEIT   SAVE     (14,12),,*
+SAVEIT   B        10(8,15)       Branch around ID
+         DC       AL1(6)
+         DC       CL6′SAVEIT′    Identifier
+         STM      14,12,12(13)   Save registers
```

VSE	RETURN	(r1,r2)
MVS	RETURN	(reg1 ,reg2) ,T ,RC = number ,RC =(15)

Under MVS, the RETURN macro returns control to the calling program and signals normal termination of the returning program. Control returns after restoring the address of the calling program's save area into register 13. The return is made by executing a branch instruction using the address in register 14. You can write the RETURN macro to restore a designated range of registers, provide the proper return code in register 15, and flag the save area used by the returning program.

Sample MVS RETURN - using 'T' Operand:

```
   L        13,4(13)        Get backward chain pointer
   RETURN   (3,6),T         (caller's save area)
+  LM       3,6,12(13)      Restore the registers
+  MVI      12(13),X'FF'    Set return indicators
+  BR       14              Return
```

Sample MVS RETURN - Using 'Return Code' Operand:

```
   LA       15,0            Set return code zero in R15
   L        13,4(13)        Get backward chain pointer
   RETURN   (14,12),RC=(15)
+  L        14,12(13,0)     Restore register 14
+  LM       0,12,20(13)     Restore registers 0 - 12
+  BR       14              Return
```

Note: You should have previously loaded a return code value into register 15.

Figure 29 on page 274 shows an example of MVS coding for initiation and termination procedures.

```
When this program received control from MVS
Reg. 13 contained address of MVS save area.
Reg. 14 contained address of MVS return.
Reg. 15 contained address of this program's entry point.

 PROGA    START
          SAVE  (14,12),,*    Store Regs in MVS save area
+         B     10(,15)
+         DC    AL1(5)
+         DC    CL5'PROGA'
+         STM   14,12,12(13)
          LR    12,15         Load start address in Reg 12
          USING PROGA,12      Define Reg 12 as base reg
          ST    13,SAVEIT+4   Store address of MVS save
 *                            area in PROGA's save area
          LA    11,SAVEIT     Load address of this Program
 *                            save area into Reg 11
          ST    11,8(13)      Store address of PROGA's save
 *                            area in MVS save area
          LR    13,11         Load Reg 13 to point to this
          .
  APPLICATION PROGRAMMER LOGIC
          .
          L     13,SAVEIT+4   Load address of MVS save area
 *                            into Reg 13
          RETURN (14,12),RC=0 Restore registers and branch
 *                            to MVS Return Address
+         L      14,12(,13)   Restore Register 14
+         LA     15,0         Load Return Code
+         LM     0,12,20(13)  Restore the registers
+         BR     14           Return
 SAVEIT   DC     18F'0'
```

Figure 29. Sample Initiation Termination Coding

13.2.1.3 Communication Region

VSE has a communication region, a storage area within the supervisor, that contains:

- The date
- The job name
- User program communication bytes
- User program switch indicators (UPSI)
- Problem program area addresses.

MVS does not provide a similar fixed area in the control program. Some of the VSE communication region facilities are available in MVS as explained in the following text.

Date

The VSE macro instruction COMRG provides the address of the communication

region in register 1. The first eight bytes of the communication region is the date in the form MM/DD//YY (month/day/year) or DD/MM/YY (day/month/year).

Job Name

The VSE communication region contains the job name that appears in the JOB control statement. This name remains for the duration of the job and can be used in a job by using the COMRG macro to get the address of the communication region and a displacement of 24 to get the job name.

User Program Communication Bytes

In VSE, the problem program can modify the communication region. You can use bytes 12 to 22 to communicate results of one job step to succeeding job steps.

MVS can transfer a return code at job completion time. The initiator/terminator, via JCL (the COND parameter), examines the code but does not pass it to the next job step. You can communicate data from one job step to the next in the same job by passing a data set from one step to another or by including a user-written SVC routine.

UPSI (User Program Switch Indicators)

UPSI consists of one byte set to binary zero when the JOB control statement is encountered. You can modify the VSE UPSI byte in two ways.

1. Through an UPSI job control statement.
2. By the problem program.

In MVS, the PARM field of the EXEC statement or a control card can be used to pass information from the JCL to the assembler program.

Problem Program Area Addresses

The VSE communication region has five fields that relate to the problem program area:

1. The address of the first byte of background problem program area.
2. The address of the uppermost byte of problem program area.
3. The address of the uppermost byte of current problem program phase.
4. The address of the uppermost byte used in loading any phase of the problem program.
5. Length of the problem program label area.

This information is generally used for two main reasons by VSE programs:

1. To dynamically expand a phase at execution time into available storage for maximum program efficiency.
2. To dynamically load phases into available storage locations and avoid overlays where possible.

Although MVS does not provide similar fields, both techniques of dynamic virtual storage utilization are available to you: an explicit request for virtual storage, and an implicit request. Using virtual storage directly by the requesting module

is an explicit request. If a separate module is requested, then the additional virtual storage requirement is implicit in that request.

The MVS GETMAIN macro is an explicit request for additional virtual storage. You can use GETMAIN to obtain a single block or multiple blocks of virtual storage. You can specify either a fixed or variable amount of virtual storage. At execution time, the macro provides information to indicate whether the virtual storage is obtained, and also to pass the address of the beginning of the block and its size (where variable block sizes are specified). Further points should be mentioned relating to explicit virtual storage requests.

1. Under MVS, you cannot take available virtual storage within an address space. MVS must allocate the virtual storage through the GETMAIN macro to the load module, even though the unused virtual storage is within the boundaries of the address space. The task will abnormally terminate if you attempt to reference unallocated virtual storage.
2. Although the allocation of blocks of virtual storage must be from within the address space, they are not necessarily contiguous.

To dynamically call modules into available virtual storage (make an implicit request), you should use the set of macros associated with dynamic program structure. When you use these macros, MVS automatically examines available virtual storage and loads the module into an open area.

13.2.1.4 Communications Region Simulation

A COMRG macro can be written to build an area similar to the VSE communication region and return the address of this area in register 1.

If your program is not going to use this information immediately, you should load the address from register 1 into another register (2-12), or store the address into a fullword in your program. This macro can obtain variable data to simulate the communications region layout in some of the following ways:

- MVS data can be communicated to a program by means of the PARM field of the EXEC statement. When control passes to your program from MVS, register 1 contains the address of a fullword on a fullword boundary in your area of virtual storage. The high order bit (bit 0) of this word is set to one. The low-order three bytes of the fullword contain the address of a two-byte length field on a halfword boundary. The length field contains a binary count of the number of bytes in the PARM field, which immediately follows the length field. If you omitted the PARM field in the EXEC statement, the count is set to zero.
- Note that if you specify numbers in the PARM parameter, they are translated into decimal digits. If you specify PARM=101, the PARM field is x′F1F0F1′.
- Data supplied in this manner is available for the duration of a single job step. If multiple job steps within a job require the data, the PARM field must be specified in each EXEC card or propagated through all EXEC cards in a job through procedure variables.
- A routine is needed to return a bit string to the byte associated with the external name UPSI. If an assembler program uses the COMRG macro and references only date, UPSI and job name as data, then no logic changes are required with this approach.
- If you use the COND parameter of the EXEC card (instead of UPSI switches) to control job flow, then the UPSI tests can be simply removed from your

programs. You can also leave the tests in your program and provide an UPSI constant in your COMRG macro that would always satisfy the tests.

- The MVS macro instruction TIME provides the system date (Julian or Gregorian) in a user work area. The date is in packed decimal digits, such as X'19980323'. For details, see the section "GETIME Macro" on page 278. If you modify the partition date with a //DATE card in the VSE JCL, you may have to simulate this option under MVS because only a single date is maintained for the entire MVS system.
- In MVS, you can obtain the job name by using the EXTRACT macro and coding the parameter FIELDS=TIOT. You obtain the address of the TIOT (task input,output table) through this macro. This job name is located in the first eight bytes of the TIOT.
- A less desirable technique would read the variable data from the operator console upon request.

If any other data is required from the COMRG macro, solutions will then have to be approached through manual procedures on an individual basis.

COMRG and MVCOM Macros

The VSE COMRG macro places the address of the communication region in register 1. The MVCOM macro modifies the user program communication bytes and the UPSI byte in the communication region. MVS does not have similar macros because it does not have a communication region (refer to the section 13.2.1.3, "Communication Region" on page 274).

LOAD Macro

The VSE LOAD macro causes the control program to bring in the phase specified in the first parameter and returns control to the calling phase. After execution of the macro, the entry-point address of the called phase is returned in register 1. The MVS LOAD macro causes the control program to bring the load module containing the specified entry point into virtual storage if a usable copy is not available in virtual storage. The entry-point address of the load module is returned in register 0.

If the application program invokes the loaded program only once, you may substitute the MVS LINK macro for the VSE LOAD and CALL macros. The LINK macro must be used to invoke the SORT utility program.

If you do not want the application program to invoke the loaded program (a table, for example), you should use the MVS LOAD macro. In addition, the application program must reconcile the fact that the VSE LOAD will return the address in register 1 and MVS LOAD in register 0.

In either case, if the VSE program used the COMRG macro to determine the load point, it is not required because MVS automatically manages load module placement in storage.

Example:

```
(VSE)
         LOAD  PROGB                LOAD the phase
         LR    15,1                 pass address
         CALL  (15),parm1,parm2     invoke PROGB
```

```
(MVS)
         LOAD  EP=PROGB               LOAD the load-module
         LR    15,0                   pass address
         CALL  (15),parm1,parm2       invoke PROGB
```

FETCH Macro

The VSE FETCH macro loads the phase specified in the first parameter and passes control to the address specified by the second. The MVS LINK and XCTL macros pass control to a specified entry point. When modifying programs from VSE to MVS, use LINK when the called phase does not overlay the calling phase. Use XCTL when the called phase does overlay the calling phase. When using XCTL, ensure that all DCBs and ACBs in the calling programs have been closed prior to issuing XCTL.

CDLOAD and CDDELETE Macros

The MVS LOAD and DELETE macros have functions similar to the VSE CDLOAD and CDDELETE macros. The CDLOAD macro can be used repetitively against the same module, first to load it, then to retrieve its address. In this case, to achieve the same result in MVS with the LOAD macro, the loaded module must be link-edited with the REUS attribute.

Example:

```
(VSE)
         LA    1,PHASENM              address of the phase name
         CDLOAD (1)                   LOAD the phase
         LR    15,1                   pass address
         CALL  (15),parm1,parm2       invoke PROGB
(MVS)
         LOAD  EPLOC=PHASENM          LOAD the load-module
         LR    15,0                   pass address
         CALL  (15),parm1,parm2       invoke PROGB
```

WTO and WTOR Macros

The MVS WTO and WTOR macros have functions similar to the VSE/ESA WTO and WTOR macros.

GETIME Macro

The VSE GETIME macro provides the time of day, (local or Greenwich Mean Time) based on a 24-hour clock, in register 1 in a form dependent upon the operand(s).

The MVS TIME macro has the same basic function as the VSE GETIME LOCAL macro. The main differences between the two macros is in register usage and degree of precision (Figure 30 on page 279). For the DEC, BIN, and TU operands, the TIME macro returns the time in register 0 and the Julian date in register 1. The date is returned in register 1 as packed decimal digits in the form 0C YY DD DF, where 0C is the century indicator, YY is the last two digits of the year, DDD is the day of the year and F is a sign character that allows the date to be unpacked and printed. If the date is needed as day/month or month/day, you must provide a routine to convert the data.

The MVS TIME macro has an additional operand MIC,address that causes the time of day to be returned in the eight-byte area specified by the address. The time of day is in microseconds, with bit 51 equivalent to one microsecond. Register 0 contains 0, and register 1 contains the date.

```
System     Operand      Register Content
VSE        STANDARD     H HM MS S
MVS        DEC          HH MM SS th
VSE        BINARY       seconds
MVS        BIN          hundredths of a second
VSE        TU           1/306 of a second units
MVS        TU           26 micro second units
H = hours          M = minutes       S = seconds
t = 0.1 seconds    h = 0.01 seconds
```

Figure 30. VSE and MVS Time Degrees of Precision

The MVS TIME macro returns the date and time into a work area when LINKAGE=SYSTEM is specified. The date (Julian or Gregorian) and the time are returned in packed decimal digits.

Example:

```
          TIME  DEC,OUTAREA,DATETYPE=YYYYMMDD,LINKAGE=SYSTEM
          .  .  .  .
OUTAREA   DS    0XL16
HHMMSSHH  DS    X'12305919'           TIME OF DAY
          DS    XL4                   filler
YYYYMMDD  DS    X'19980323'           DATE
          DS    XL4                   filler
```

PDUMP Macro

The VSE PDUMP macro provides a hexadecimal dump of the general registers and of the main/virtual storage area contained between the two address parameters. The output is automatically written on the device assigned to SYSLST with 121-byte records.

The MVS SNAP macro provides you with the same dump facility. However, you must supply the data set (via the DD statement) on which the output is written and a DCB for the data set must be opened before the SNAP macro is used.

VSE	PDUMP	address1 , address 2 ,MFG=area r1 r2 (S,area) r3
MVS	SNAP	DCB = dcbaddress ,TCB = address (2-12) (2-12) ,ID= number ,SDATA = (sysda—a) (2-12) ,PDATA = (probda—a) ,STORAGE = (s—ar— address,end address) (2-12) (2-12) ,STRHDR = (hdr addr) hdr lis— addr ,LIST = address of lis— (2-12)
No—e: TCB = address is used wi—h sub—asking (2-12)		

DUMP Macro

VSE	JDUMP DUMP	RC=
MVS	ABEND	comple—ion code (1-12) ,DUMP ,STEP
No—e: Use ,STEP wi—h sub—asking.		

The VSE DUMP macro and the VSE JDUMP macro terminate the job step and give a hexadecimal dump of the general registers, supervisor and the partition that issued the macro if the main program or task made the request. If a subtask issues the macro, the subtask is detached, but the partition is not terminated. The dump is directed to SYSLST.

The MVS ABEND macro causes the termination of the current job step. A dump of all virtual storage areas, control blocks, and the trace table is recorded if you provide a //SYSUDUMP DD statement. In a multitasking environment, the task that issues the macro with its subtasks is terminated. If the parameter STEP is included, the entire job step is terminated. The remaining job steps are either skipped or executed, depending on the parameters in the COND entry in the JOB and EXEC statements.

CANCEL Macro

VSE	CANCEL	ALL
MVS	ABEND	comple—ion code (1-12) ,DUMP ,STEP
No—e: Use .STEP wi—h sub—asking.		

The VSE CANCEL macro causes the termination of the job. All succeeding job steps within this job are automatically bypassed by job control.

The MVS ABEND macro provides this same facility. Refer to the section “DUMP Macro” on page 280.

EOJ Macro

VSE	EOJ	RC=
MVS	RETURN	(reg1 ,reg2) ,T ,RC=number ,RC=(15)

The VSE EOJ macro allows you to terminate a problem program step. Any routine of this problem program can issue it.

The MVS RETURN macro returns control to the calling program and signals normal termination of the returning program. To terminate a job step, you must issue this macro by a routine at the same level as the one originally called to begin execution of the job step. See 13.2.1.2, “Termination” on page 269 and note Register 13 load requirements.

LOCK and UNLOCK Macros

The MVS ENQ and DEQ macros have functions similar to the VSE LOCK and UNLOCK macros.

Example:

```
(VSE)
         LOCK  DTL1                    Get the lock
         .  .  .  .
         UNLOCK DTL1                   Free the lock
         .  .  .  .
DTL1     DTL   NAME=L20,CONTROL=E,LOCKOPT=1,SCOPE=EXT
(MVS)
         ENQ   MF=(E,DTL1)             Get the lock
         .  .  .  .
         DEQ   MF=(E,DTL1)             Free the lock
         .  .  .  .
DTL1     ENQ   (QNAME1,RNAME1,E,,SYSTEM),MF=L
QNAME1   DC    CL8'VSELOCK'
RNAME1   DC    CL12'L20'
```

CHKPT Macro

VSE	CHKPT	res–ar– address SYSnnn, (r1) , end address , –poin–er (r2) (r3) dpoin–er , filename (r4) (r5)
MVS	CHKPT	dcbaddress ,checkid address ,checkid leng–h ,'S' CANCEL

The MVS CHKPT macro is similar to the VSE CHKPT macro with two minor differences in the checkpoint logic. One difference is in the use of registers when control returns to you after the CHKPT macro. In VSE, register 0 indicates successful or unsuccessful checkpointing. In MVS, register 15 indicates not only successful or unsuccessful checkpointing, but also successful restart. Another difference is in the restart logic. In VSE, you specify a restart address as one of the parameters in the CHKPT macro. MVS automatically restarts with the instruction following the CHKPT macro.

Many VSE parameters are not necessary in MVS:.

- The SYSnnn parameter in VSE specifies the logical unit on which the checkpoint is stored. The dcbaddress parameter of the MVS CHKPT macro gives the address of the user-coded data control block (equivalent to a DTF) for the checkpoint data set, which must be a magnetic tape or direct access volume.
- A restart address is not given in MVS.
- An end address used in VSE to specify the highest storage address to be dumped during CHKPT is not needed, because MVS automatically dumps the entire contents of the program's virtual storage data areas.
- *tpointer* in VSE provides a list to the CHKPT macro of the tape files used in the program. If this parameter was not specified, repositioning of tape files would not be performed at restart. MVS automatically repositions tape files without the use of a similar parameter.
- *dpointer* in VSE permits operator volume verification at restart time. The facility is provided by MVS when it allocates devices for a particular job step.
- Filename in VSE points to the DTF describing the disk CHKPT file. This file must be opened prior to use, and label checking is performed at that time. The MVS DCB is roughly equivalent. The data set may be opened by you or by the checkpoint routine when the CHKPT macro is executed. It is recommended that the user issue the OPEN rather than default to the system, because the system opens every time a CHKPT is issued.

13.2.2 Multitasking Macros

Under VSE, when you specify asynchronous processing at system generation time, the multitasking group of macros is supported to permit more than one task to execute within each partition. Each subtask must be initiated by the main task: control then passes to the subtask.

The storage protection key and priority of the partition remain the same, but the priority of a task within a partition is determined by the sequence of subtasks it attached. Thus, if subtask 1 through subtask n are attached in ascending order, the priority of execution within the partition would be from subtask 1 to subtaskn, then to the main task.

Under MVS, asynchronous processing is supported. This dynamic parallel structure differs from VSE in that:

- Subtasks may be created by job step tasks or other subtasks.
- No absolute rules exist for assigning priorities to tasks and subtasks. When created, subtasks are assigned a priority by the originating task. Within limits, this priority may be higher, lower, or the same as the originating task.
- Any time during execution, the originating task may modify the priority of an attached subtask (as long as it has the task control block (TCB) address of the subtask).

The VSE multitasking macros can be divided into three general categories:

- Subtask initiation and normal termination macros ATTACH/DETACH.
- Resource protection macros RCB/ENQ/DEQ.
- Intertask communication macros WAITM/POST.

The MVS counterparts and their comparable features follow.

13.2.2.1 ATTACH/DETACH Macros

VSE	ATTACH	entrypoint|(S,entrypoint)|(r1) ,SAVE=savearea|(S,savearea)|(r2) ,ABSAVE=savearea1 (S,savearea)&vbar,(r3) ,ECB=ecbname|(S,ecbname)|(r4) ,MFG=area|(S,area)|(r5) ,RETURN=NO|YES ,NAME=(name(S,name|(r8))
MVS	ATTACH	EP=symbol ,DCB=dcb address EPLOC=address of name (2-12) (2-12) DE=address of list entry (2-12) ,LPMOD=number (2-12) ,DPMOD=number (2-12) ,PARAM=(address ,...) (2-12) ,VL=1 ,ECB=ecb address (2-12) ,ETXR=exit routine (2-12)

ENTRYPOINT

In VSE, it defines the storage address of the entry point of the subtask. The entry point must be in storage before the subtask can be successfully attached. The EP, EPLOC or DE parameter in MVS causes the required module to be loaded into storage (if it is not already in storage) and begins execution at the entry specified.

The entry-point name must be a member name or an alias in a directory of a partitioned data set, or it must have been specified in the IDENTIFY macro. If the specified entry point cannot be located, the new subtask is abnormally terminated. MVS requires the subtask to perform normal initialization and termination coding. Therefore, the MVS subtasks are generally separate load modules rather than interspersed coding that commonly is found in VSE subtasks.

SAVE: Defines the address of a formatted user save area for the subtask containing the general purpose (and floating point) registers while the VSE task is not active. MVS assumes the responsibility for providing areas to save registers.

ECB: Defines a task event control block used in intertask communications and task synchronization. Both operating systems use a fullword ECB, but individual bits have different meanings.

ABSAVE: Used if the subtask is to execute the main task's abnormal termination STXIT AB routine under VSE. When an ABEND is issued in MVS for a task that has previously issued an STAE macro, the ABEND is intercepted and control is given at the STAE exit routine address.

An additional facility available under MVS is the ability to specify an end-of-task exit routine (ETXR) to be given control after the new task is normally or abnormally terminated. This is true even if the originating task was active and must be in virtual storage when required. In a sense, this facility functions something like an STXIT routine. When a termination interrupt occurs, the routine is given control. This is a useful facility that you should examine when converting the ATTACH macro.

VSE	DETACH	SAVE=savearea (1)
MVS	DETACH	—cb loca—ion address (1-12)

The VSE DETACH macro, issued by the main task or the subtask itself, causes the subtask to terminate. An EOJ or CANCEL macro can also perform this function.

Under MVS, the DETACH macro is used to remove the task control block of the subtask from the system and to terminate the subtask and all its subtasks. DETACH is required when either ECB or ETXR was specified in the ATTACH for this subtask. Both of these parameters signal to the initiating task that the subtask has terminated normally or abnormally. The task control blocks for all subtasks must be removed by the originating task before the originating task can

terminate normally. MVS does not permit the subtask to issue its own DETACH. If neither ECB nor ETXR is specified in the ATTACH, the subtask is removed from the system automatically at normal termination. In this case, no DETACH should be issued.

13.2.2.2 WAIT/POST Macros

The two operating systems provide macros that synchronize task execution if one task or subtask depends upon the completion of another subtask.

Under VSE and MVS, the WAITM and WAIT macros, respectively, inform the control program that the execution of an active task cannot continue until one or more specific events, each represented by a different control block, have occurred.

The POST macro signals completion of an event. A POST issued to an (ECB) removes from the wait state a task waiting for the event to complete.

VSE	POST	ecbname ,SAVE= savearea (1) (0)
MVS	POST	ecbaddress ,comple—ion code (1-12) (2-12) (0)

ECBNAME: Provides the address of the particular event control block (ECB) representing the event posted as complete. The MVS ecbaddress parameter is the equivalent.

SAVE: If this operand is present, only the task identified by the address of its save area is taken out of the wait state. Although time is saved when specifying this operand, other tasks waiting for this ECB are not taken out of the wait state for this event until another POST is issued.

When a POST is issued without the SAVE operand, all tasks waiting for the ECB are taken out of the wait state, and the highest-priority task regains control. You can use the completion code parameter of the MVS POST macro to pass information to the waiting subtask or tasks. These tasks can then interrogate the code set in the ECB to determine a continuation of the wait or a return to execution.

VSE	WAIT	ecb1,ecb2,... lis—name (1)
MVS	WAIT	number of even—s,ECB = address (2-12) (1-12) (0) ECBLIST=address (1-12)

The systems provide different facilities. Because of the use of ECB bits under MVS, only one WAIT macro can refer to an ECB at one time. An additional MVS facility, specified through the number of events parameter, permits the task to be taken out of the wait state after the specified number of events has been posted

as complete. This number may be less than or equal to the number of ECBs specified in the macro.

13.2.2.3 RCB/ENQ/DEQ Macros

This set of macros enables you to protect data files or other resources when processing a multitasking environment.

The VSE RCB macro generates an aligned doubleword resource control block that functions much like an ECB. The VSE ENQ and DEQ macros test the status of the resource through the RCB name.

MVS does not require or use an RCB macro. MVS generates an entry, within the supervisor, which is used in a control area by the ENQ and DEQ macros to test the resource status. Because the MVS entries are stored, tested, and modified within the supervisor, you can protect resources across address spaces. VSE can only protect resources within a partition because the area used to contain the resource status is within the application program area.

VSE	ENQ	rcbname (0)
MVS	ENQ	qname address , rname address, E (2-12) (2-12) S rname leng–h , SYSTEM ,... (2-12) STEP SYSTEMS

Under VSE you can specify only one resource in each ENQ.

The MVS ENQ macro offers many additional facilities. The ENQ macro requests the control program to assign control of one or more serially reusable resources to the active task. If any of the resources are not available, the active task is placed in the wait state until all of the requested resources are available. The qname and rname parameters, roughly equivalent to the VSE rcb name, represent names of a common resource to the control program. These names may or may not have any relation to the actual name of the resources.

The control program does not associate the name with the actual resource. It merely processes requests having the same qname and rname on a first-in, first-out basis. It is your responsibility to associate these names with the actual resource. These parameters may define a resource name pertaining to this step only, or a resource name that may be denoted throughout the entire system.

VSE	DEQ	rcbname (0)
MVS	DEQ	qname address , rname address (2-12) (2-12) rname leng–h , STEP ,... (2-12) SYSTEM SYSTEMS

The parameters and additional facilities available under MVS are similar to those explained under the ENQ macro.

13.2.3 Interrupt Handling Routines

Interrupt routines take care of the interval timer, abnormal conditions, and operator communication interrupts. Abnormal condition interrupts will not be addressed in this publication.

13.2.3.1 Interval Timer Interrupts

Basically, the two ways of utilizing the interval timer in your programs are routine handling and wait handling.

Routine Handling

This method allows a branch to your own routine when the interval timer interrupt occurs. The following illustrates the VSE macros and corresponding MVS macro needed to perform this operation:

VSE	STXIT	IT, rou—ine address , save area (0) (1)
	SETIME	seconds (1)
	EXIT	IT
MVS	STIMER	REAL , —imer comple—ion exi— address TASK (2-12) WAIT ,DINTVL = address (2-12) ,BINTVL = address (2-12) ,TUNINTVL=address (2-12) ,TOD = address (2-12)

The VSE macro instructions allow you to execute a routine when a timer interrupt occurs and then to return to the program that was being executed prior to the interruption. The STXIT macro provides the address of the routine to be executed. A register save area must also be specified in the STXIT macro. The saving and restoring of the registers are performed automatically by the supervisor. The SETIME macro sets the interval timer. The EXIT macro returns from the specified routine (STXIT macro) to the point in the interrupted program where the interruption occurred.

The single MVS macro (STIMER) also allows you to execute a routine when the timer interrupt occurs. The REAL parameter indicates that time is to be decreased continually, while TASK means that the timer is stopped or adjusted only when the task is active. WAIT means that the job step is to be placed in a wait state until the interval expires. The timer completion exit address parameter is the same as the routine address used in the VSE STXIT macro.

The DINTVL parameter indicates that the time interval requested is in decimal units and is stored in the address specified. The MVS timer routine must follow the standard conventions for handling registers.

You must save the registers on initiation and restore them at routine termination.

Wait Handling

This method of using the interval timer allows you to set the timer and wait for the time to elapse. The job or task is prevented from executing until the interrupt occurs.

VSE	TECB SETIME WAIT	 seconds , –ecbname (1) (2-12) –ecbname (1)
MVS	STIMER	WAIT ,DINTVL = address (2-12) ,BINTVL = address (2-12) ,TUINTVL= address (2-12) ,TOD = address (2-12)

The VSE TECB is an event control block that records the status of the timer. When the interrupt is received, this condition is posted in the specified TECB. The WAIT macro tests the TECB status and determines if the interrupt has occurred.

The MVS STIMER macro provides basically the same facility. The difference is that, under VSE, you can insert code between the point where the timer is initiated (SETIME) and where the test for the interrupt is performed (WAIT). The wait is automatically built into the MVS STIMER macro. The WAIT parameter indicates that you do not want the coding below the macro to be executed during the time period. The DINTVL parameter is explained under “Routine Handling” on page 287.

TTIMER Macro

The VSE and MVS TTIMER macros are compatible with each other:

VSE	TTIMER	CANCEL
MVS	TTIMER	CANCEL ,TU

The VSE time interval is expressed in binary in hundredths of a second, while under MVS, the interval is expressed in binary in timer units (one timer unit equals 26.04166 microseconds).

13.2.3.2 Operator Communication Interrupts

VSE	STXIT EXIT	OC, rou–ine address , savearea (0) (1) OC

The preceding combination of VSE macro instructions allows you to execute a routine when an operator attention interrupt occurs and to return to the program that was being executed before the interruption.

MVS has no equivalent function. It is possible, however, to simulate this function by using the WTOR macro. This MVS macro writes a message requiring a reply on the operator console and provides the information required by the control program to relay the reply to the issuing program. You must wait for the reply.

This macro could also be executed without a following WAIT macro and coding inserted in the program loop to test the completion of the event. Upon completion, it is possible to execute the communication routine and later resume normal processing. Another solution is to include the communication routine as a subtask with the WTOR macro instruction followed by a WAIT instruction.

Alternately, you can use the Job Control PARM Field (see Register 1 in “Register Conventions” on page 269) to pass information to the program. You may realize throughput improvements because there is no wait for operator′s reply.

Under VSE, you can initiate communication with the background partition using the interrupt key on the console. You can use the MSG (message) command to initiate communication with a foreground partition.

Under MVS, the operator interfaces with job management to provide operator-to-system communication via commands entered on the operator console. This allows the operator to respond to requests from processing programs. However, operator-to-processing program communication must be initiated internally by the processing program. There is no support of the external interrupt key for operator-to-program communication; it switches from the primary to the alternate console.

13.2.4 Virtual Storage Macros

13.2.4.1 GETVIS and FREEVIS Macros

The MVS GETMAIN, FREEMAIN and STORAGE macros have functions similar to the VSE GETVIS and FREEVIS macros. Example:

(VSE)

```
         GETVIS LENGTH=1000,ADDRESS=PTR1
         LTR   R15,R15                GETVIS OK?
         BNZ   ERROR1                 NO, CANCEL
         .  .  .  .
         FREEVIS LENGTH=1000,ADDRESS=PTR1
         .  .  .  .
PTR1     DS    A
```

(MVS)

```
         GETMAIN RC,LV=1000
         LTR   R15,R15                GETVIS OK?
         BNZ   ERROR1                 NO, ABEND
         ST    R1,PTR1                save storage address
         .  .  .  .
         L     R1,PTR1                pick up storage address
         FREEMAIN RC,LV=1000,ADDRESS=(1)
         .  .  .  .
PTR1     DS    A
```

13.2.4.2 RELPAG Macro

The MVS PGRLSE and PGSER RELEASE macros have functions similar to the VSE RELPAG macro.

VSE	RELPAG	begin addr , end addr,777 (2-12) (2-12) lis—name (1)
MVS	PGRLSE	LA=addrl ,HA=addr2 (2-12) (2-12) (0) (1)

When you issue the PGRLSE macro, all complete pages of virtual storage between the low and high addresses specified are released. You can help reduce system overhead by releasing virtual storage when you no longer need it.

Unlike VSE, you must issue a separate instruction for each area of storage you want to release.

13.2.4.3 PFIX and PFREE Macros

The MVS PGFIX and PGFREE macros correspond to VSE PFIX and PFREE. Use of the MVS instructions is restricted, however, to system functions and authorized users (key-0 and supervisor state). Further information about PFIX and PFREE appears in MVS SRL library publications.

13.2.4.4 SETPFA Macro

In MVS, only system functions and authorized users (key-0 and supervisor state) are permitted to handle their own page faults.

13.2.4.5 PAGEIN Macro

The MVS PGLOAD macro is similar to PAGEIN, but its use is restricted to system functions and authorized users (key-0 and supervisor state).

13.2.4.6 FCEPGOUT, RUNMODE, VIRTAD and REALAD Macros

These instructions have no equivalent in MVS.

13.2.5 VSAM Macros

Detailed information on the coding of MVS VSAM macros can be found in the publication *DFSMS/MVS Macro Instructions For Data Sets*, SC26-4913.

13.2.5.1 ACB Macro

The ACB macro is source-compatible except for the parameter ″PARMS=″ which was introduced by VSE VSAM Release 2 to support VSE VSAM Space Managed files. MVS VSAM does not support this type of file. Programs using this VSE feature should be converted to use MVS Sequential Access Method (SAM), or VSAM ESDS files.

Additional MVS VSAM ACB Parameters

- **CATALOG=YES/NO** - NO is used if the catalog is to be processed as a normal cluster with normal GET/PUT macros. Programs must be APF-authorized to process a catalog as a data set.
- **MACRF=(**
  - *ICI* - Improved-Control-Internal-Processing (ICIP) is to be used. ICIP is a VSAM ″fast-path″ that reduces CPU utilization. However, the functions that can be used are severely restricted. See the *VSAM Administration Guide* for more details.
  - *CFX* - Control blocks and buffers are fixed in real storage. ICI must also be specified.
  - *DSN* - Data Set Name sharing specifies that the basis for sharing control blocks and buffers is by matching VSAM NAMEs when the files are opened by the ***SAME*** task. Unless this option is used, a separate ACB that opens the same data set can have the same integrity problems as a program within another region. This includes potential destruction of the file if concurrent updates are allowed. DSN is most commonly used to tie buffers of a base ACB to the control blocks of the base ACB associated with a path.
  - *SIS* - Sequential Insert Strategy is used to override the default control interval or area split algorithm for direct processing. SIS will cause the splits to occur at the insert point rather than at midpoint when direct PUTs are done. Although positioning is lost and writes are done after each direct PUT request, SIS allows more efficient space usage when direct PUTs are done against ascending keys in a KSDS.
  - *GSR* - Global Shared Resources specifies that the data set is to be tied to a common VSAM resource pool of control blocks, strings and buffers that is allocated in the MVS Common System Area (CSA). This provides for VSCR.
  - *LSR* - Local Shared Resources is similar to GSR except the resource pool is built in a single address space and integrity is limited to that region.
- **BSTRNO = n** - The number of strings allocated to the base cluster associated with a Path ACB. For a Path ACB, the default is STRNO; however, if you are using DSN sharing to tie a separate base ACB control block structure to the path, you can specify additional base strings for that purpose. Refer back to ″DSN″ for why you want to use DSNAME sharing.

13.2.5.2 EXLST Macro and EXCPAD Routines

A VSE VSAM EXLST macro that has a EXCPAD routine address coded will have to be converted. There is not an equivalent exit routine in MVS VSAM. The closest equivalent is the MVS VSAM UPAD exit routine which allows user processing during a VSAM request. However, the GENCB, MODCB, SHOWCB, and TESTCB macros do not support the UPAD exit routine. Great care should be taken in converting and testing a converted EXCPAD routine, since it is, generally, very complicated code.

13.2.5.3 RPL Macro (Additional MVS Parameters)

- **ECB=address** - Request to VSAM to post the specified ECB at the completion of the RPL request.
- **MSGAREA=address** - In the case of a physical error, VSAM will place a message in the area address specified.

- **MSGLEN=length** - The length of the MSGAREA specified above. The size of a message is 128 bytes.
- **OPTCD=(**
  - *ASY* - Asynchronous access; VSAM returns control to the program after scheduling the request so the program can do other processing while the request is earned out. SYN is the default and specifies that VSAM will return to the processing program when the request is complete.
  - *WAITX* - If ″OPTCD=(..,SYN..)″ and the specifics ″MACRF=(..,LSR or GSR..)″, ***and*** ″EXLST ...,UPAD=″, then VSAM branches to the UPAD routine address when VSAM would otherwise have issued a WAIT. The most common use of UPAD routines is by IMS/VS or it may be used as a replacement for the VSE VSAM EXCPAD routines.

13.2.5.4 SHOWCB Macro

The VSE VSAM SHOWCB macro may not be source compatible depending on the fields requested. The use of the macro and the results should be carefully examined.

MVS VSAM Additional SHOWCB Fields

- **BFRFND** - Number of successful GETs without physical reads required. (LSR or GSR only)
- **BUFRDS** - Number of GETs requiring physical reads. (LSR or GSR only)
- **ENDRBA** - Ending RBA of the space used by the data or index components (that is, the last byte used in the component).
- **HALCRBA** - The High-Allocated-RBA of the specified component.
- **NUIV** - Number of physical WRITEs not issued by the user. (LSR or GSR only)
- **UIW** - Number of WRITEs issued by the user. (LSR or GSR only)

13.2.5.5 MVS VSAM CHECK Macro

The MVS VSAM CHECK macro is used to suspend processing until VSAM has completed the request associated with the RPL. This macro is used for asynchronous processing (that is, OPTCD=(ASY..).

13.2.5.6 VSE VSAM TCLOSE Macro

The TCLOSE macro of VSE is coded in MVS: ″CLOSE ...,TYPE=T″

13.2.5.7 VSAM Error and Reason Code Compatibility

MVS VSAM error codes and reason codes may have a slightly different meaning than VSE VSAM. In all cases, MVS documentation should be consulted. Especially in Assembler, the logic for specific error codes should be verified. In some cases, MVS VSAM provides additional information.

13.2.6 Data Management Macros

This section compares the VSE and MVS data management macro instructions. Detailed information on the coding of MVS data management macro operands can be found in the publication *DFSMS/MVS Macro Instructions For Data Sets*, SC26-4913.

Note: As a VSE user, you are familiar with the term filename. In MVS, filename is referred to as dcbaddress.

13.2.6.1 List and Execute Macro Forms

The list and execute forms of data management macro instructions, used together provide the same services available from the standard form of the macro. The list form of the macro provides a parameter list to be passed to either the control program or another problem program, depending on the micro instruction.

Use the execute form with one or two parameter lists established by the list form. The execute form provides the executable instructions required to modify the parameter lists and pass control to the required program. The advantages of the list and execute forms of a macro are:

- Any operands that remain constant in every use of the macro can be coded in the list form. You can omit these operands in each of the execute forms of the macro that uses the list. This saves coding time and virtual storage area when you use a macro many times.
- The execute form of the macro can modify any of the operands previously designated. There are some exceptions.
- The list used by the execute form of the macro can be located in a portion of virtual storage assigned to the task through the use of the GETMAIN macro. This ensures that the program remains reentrant.

13.2.6.2 Definition of BLKSIZE

In VSE, the keyword BLKSIZE is defined as the length of one input/output area. The value specified in BLKSIZE depends on the type of processing intended. Depending on the selected options in the DTF, the value specified in BLKSIZE for the VSE file can be:

- Data length (for direct access and all sequential files)
- Data length, plus key length (for direct access files)
- Data length, plus eight bytes for count (for direct access and sequential DASD files)
- Data length, plus count, plus key length (for direct access files). In all cases MVS uses the keyword BLKSIZE to mean only data length.

Note: In MVS, the maximum value for BLKSIZE (that is, length of data blocks) is 32760. In VSE, the length of data blocks on DASD or tape can be greater than 32760.

13.2.6.3 IOREG

In VSE, specification of the IOREG parameter on a DTF allows blocked records to be processed directly in the buffer. In MVS, the equivalent technique is called Locate Mode. There are two major differences between IOREG and Locate Mode:

- Locate Mode always returns the record address in R1; consequently, an LR ioreg,R1 instruction must be inserted after each GET instruction when converting to MVS.
- The combination IOREG=(reg),TYPEFLE=OUTPUT has no exact equivalent in MVS. The easiest way to convert this type of file is to use Move Mode (MACRF=PM) instead of Locate Mode (MACRF=PL).

13.2.6.4 I/O Error Checking

When an input/output error occurs under MVS, a user-written synchronous error routine (SYNAD) can be given control. You can use this routine to analyze exceptional conditions or uncorrectable errors. The error can be skipped or accepted, or processing can be terminated.

If an input/output error occurs during data transmission, standard error recovery procedures, provided by the operating system, attempt to correct the error before returning control to your program. An uncorrectable error usually causes an abnormal termination of the task. However, if you specify in the DCB macro the address of an error analysis routine, the routine receives control in the event of an uncorrectable error.

You can write an SYNAD routine to determine the cause and type of error that occurred by examining:

- The contents of the general registers.
- The data event control block.
- The exceptional conditional code.
- The standard status and sense indicators.

Use a special macro instruction, SYNADAF, to perform these functions automatically. SYNADAF produces a descriptive error message that can be printed by a subsequent PUT or WRITE macro.

Having completed the analysis, you can return control to the operating system or close the erroneous data set and terminate processing. In no case can you attempt to reread or rewrite the record because the system has already attempted to recover from the error.

When you use GET/PUT macro instructions to process a sequential data set, the operating system provides three automatic error options (EROPT) to be used if there is no SYNAD routine, or if you want to return control to your program from the SYNAD routine:

- ACC - accept the erroneous block.
- SKP - skip the erroneous block.
- ABE - abnormally terminate the task.

These options are applicable only to data errors, because control errors result in abnormal termination of the task. Data errors affect only the validity of a block of data. Control errors affect information or operations necessary for continued processing of the data set. These options are not applicable to output errors, except printer output errors. When chained scheduling is used, the SKP option is not available. If it is coded, it defaults to the ACC option. If you do not complete the EROPT and SYNAD fields, the system assumes ABE.

13.2.6.5 LIOCS Card File Definition

The methods of processing cards files are the same in VSE and MVS. Figure 31 on page 295 compares the VSE DTFCD and the MVS DCB operands. Figure 32 on page 295 gives an example of the macros that process a card file under both operating systems. Figure 33 on page 296 shows a short sample program.

```
VSE DTFCD                    MVS DCB DSORG=PS

 DEVADDR = SYSxxx            DDname (in DD statement)
 IOAREA1 = xxxxxxxx          BUFNO = 1
     or
 IOAREA1 = xxxxxxxx          BUFNO = 2 or more
 IOAREA2 = xxxxxxxx
 ASOCFLE = xxxxxxxx          UNIT=AFF=ddname (in DD statement)
 BLKSIZE = nnn               BLKSIZE = nn
 CONTROL = YES               MACRF = (..C..) for input only
 CTLCHR = YES                RECFM = (...M)
          ASA                        (...A)
 SSELECT = n                 DEVD = ..,..,STACK=1

 DEVICE = nnnn               UNIT = nnnn (in DD statement)
 EOFADDR = xxxxxxxx          EODAD = xxxxxxxx
 EAROPI = xxxxxxxx           SYNAD = xxxxxxxx
 FUNC = xxx                  DEVD = (..,..,..,FUNC=xxxxxxxx)
 IOREG =(r)                  MACRF -(...L..)
 MODE = E O                  DEVD = (..,MODE=E O
        C R                                 C R
 RECFORM = xxxxxx            RECFM = xxx
 RECSIZE = (r)               LRECL = nn
 SEPASMB = YES               User must code the DCB
 TYPEFLE = INPUT             MACRF = (G..)
           OUTPUT                    (P..)
           CMBND
 WORKA = YES                 MACRF = (...M..)
```

Figure 31. Comparison of the DTFCD and DCB Macros

```
         OPEN  CARD
VSE      GET   CARD,WORK
           .
         CLOSE CARD
CARD     DTFCD DEVADDR=SYSIPT,IOAREA1=CARDIN1,                          C
               IOAREA2=CARDIN2,EOFADDR=END,                             C
               WORKA=YES
         CDMOD WORKA=YES


         OPEN  CARD
MVS      GET   CARD,WORK
           .
         CLOSE CARD
CARD     DCB   DSORG=PS,MACRF=(GM),                                     C
               DDNAME=SYSIPT,EODAD=END,                                 C
               RECFM=FB,LRECL=80
```

Figure 32. Card File Macros in VSE and MVS

```
         OPEN  CARD
VSE      GET   CARD,WORK
           .
         CLOSE CARD
CARD     DTFCD DEVADDR=SYSIPT,IOAREA1=CARDIN1,                    C
               IOAREA2=CARDIN2,EOFADDR=END,                       C
               WORKA=YES
         CDMOD WORKA=YES


         OPEN  CARD
MVS      GET   CARD,WORK
           .
         CLOSE CARD
CARD     DCB   DSORG=PS,MACRF=(GM),                               C
               DDNAME=SYSIPT,EODAD=END,                           C
               RECFM=FB,LRECL=80
```

Figure 33. Card File Programs in VSE and MVS

13.2.6.6 LIOCS Printer File Definition

The methods of processing printer files are the same in VSE and MVS. Differences exist only in the CNTRL and PRTOV macros.

CNTRL Macro

VSE	CNTRL	filename ,code ,nl ,n2 (1)
MVS	CNTRL	dcbaddress, SP ,nl SK

Notes:

1. MVS does not support delayed printer control.
2. The MVS Job Entry Sub-system supports machine and ASCII control characters.

PRTOV Macro

VSE	PRTOV	filename , 12 , routine name (1) 9 (0)
MVS	PRTOV	dcbaddress , 12 , overflow exit address (2-12) 9 (2-12)

The PRTOV macro results in the same action in VSE and MVS. However, if you use CNTRL, you cannot use SYSOUT=A and must specify the printer in the UNIT parameter of the output data set DD statement. Since this is rarely a viable option, programs that use the PRTOV macro should be converted to use a line counter instead.

Figure 34 on page 297 shows a comparison of the DTFPR and DCB operands.

VSE DTFPR	MVS DCB DSORG=PS
DEVADDR = SYSxxx	DDname (in DD statement)
IOAREA1 = xxxxxxxx	BUFNO = 1,MACRF =(..M..)
or	
IOAREA1 = xxxxxxxx	BUFNO = 2 or more
1OAREA2 = xxxxxxxx	
ASOCFLE = xxxxxxxx	UNIT=AFF=ddname (in DD statement)
BIKSIZE = nnn	BLKSIZE = nnn
CONTROL = YES	MACRF = (PC)
CTLCHR = YES	RECFM = (..A)
ASA	(...M)
DEVICE = nnnn	UNIT = (in DD statement) (1)
ERROPT = xxxxxxxx	SYNAD = xxxxxxxx
FUNC = xxxx	DEVD=(..,..,..,FUNC = xxxxxx)
IOREG =(r)	MACRF= (..L..)
PRINTOV = YES	Use PRTOV
RECFORM = xxxxxx	RECFM = xxx
RECSIZE = (r)	LRECL = nn (2)
SEPASMB = YES	User must code the DCB
WORKA = YES	MACRF = (..M..)

1. By specifying UNIT= you are using dedicated I/O. A preferable alternative is to use SYSOUT=A.
2. For output of undefined records, you must provide the length of the records in a two-byte field (DCBLRECL) within the DCB. LRECL must be provided for fixed-length and variable-length records.

Figure 34. Comparison of the DTFPR and DCB Macros

13.2.6.7 LIOCS Tape File Definition

In VSE and MVS, you process sequential tape files in the same way.

OPEN Macro

VSE	OPEN(R)	filename ,... (rl)
MVS	OPEN	dcbaddress , (options) .... (2-12)

Options

	Option 1	Option 2
QSAM	INPUT	,REREAD
	OUTPUT	,LEAVE
	RDBACK	,DISP
	INPUT	,REREAD
	OUTPUT	,LEAVE
BSAM	RDBACK	,DISP
	INOUT	
	OUTIN	

Note: You can specify any number of dcbaddresses and associated options in the OPEN macro instruction.

CLOSE Macro

VSE	CLOSE(R)	filename ,... (r1)
MVS	CLOSE	dcbaddress ,option ,... ,TYPE=T (2-12)

1. Options

   REREAD
   LEAVE
   REWIND
   DISP

2. If you omit the option, the following positioning occurs:

   If TYPE=T is coded, LEAVE is assumed (BSAM only).
   If TYPE=T is not coded, DISP Is assumed (BSAM only).

3. You can code CLOSE with TYPE=T to temporarily close sequential data sets on magnetic tape volumes processed with BSAM. When you use TYPE=T, the DCB used to process the data set maintains its open status, and you don′t have to issue another OPEN macro to continue processing the same data set. A request to temporarily close a data set causes MVS to process labels, modify some of the fields in the system control blocks for that data set, and reposition the volume (or current volume in the case of multivolume data sets) in much the same way that the normal CLOSE macro does. When you code TYPE=T, you can specify that the volume either be positioned at the end of data (the LEAVE option) or be repositioned at the beginning of data (the REREAD option). Magnetic tape volumes are repositioned either immediately before the first data record or immediately after the last data record; the presence of tape labels has no effect on repositioning. When a DCB is shared among multiple tasks, the task that opened the data set must also close it; however, a subtask of the task that opened the DCB can issue the CLOSE macro with the TYPE=T option.

4. When using QSAM, close all output data sets before ending the program to ensure that all records have been written.

CNTRL Macro

VSE	CNTRL	filename ,code (1)
MVS	CNTRL	dcbaddress,code ,number of blocks

Notes:

1. The codes are as follows:

VSE	MVS (BSAM only)
REW RUN	No equivalen—. The op—ion specified in —he DISP parame—er of —he DD s—a—emen— is —aken. Refer also —o —he OPEN/CLOSE op—ions.
BSR FSR	BSR, number of blocks FSR, number of blocks
BSF FSF	BSM Bo—h of —hese codes cause spacing pas— —apemark, FSM —hen, spacing in —he opposi—e direc—ion over —apemark.
WTM	No equivalen—
ERG	
BSL FSL	No equivalen—

2. Under MVS, if the forward or backspace operation does not complete successfully, control is passed to the error analysis routine (SYSAD). If you do not specify a SYNAD routine, the task terminates abnormally.
3. If a tapemark is encountered for DSR or FSR, control is returned to the processing program, and register 15 contains a count of the uncompleted forward spaces or backspaces. If the operation completes normally, register 15 contains zero.
4. If you specify OPTCD=H in the data control block, the CNTRL macro instruction can perform record positioning on VSE tapes that contain embedded checkpoint records. Imbedded checkpoint records encountered during the record positioning are bypassed and are not counted as blocks spaced over. You must also specify OPTCD=H in a JCL DD statement. The CNTRL macro instruction cannot be used to backspace VSE 7-track tapes that are written in data convert mode that contain embedded checkpoint records (BSAM).

NOTE Macro

VSE	NOTE	filename (1)
MVS	NOTE	dcbaddress (1-12)

The result of the NOTE macro is the same in VSE and MVS. Information is returned in register 1 in the format 0bbb. If you use NOTE under MVS, the tape on which the data set resides must have standard labels if the OPEN option RDBACK or DISP=MOD has been specified. The MVS NOTE macro is valid only for BSAM and BPAM.

POINTW / POINTR Macros

VSE	POINTW POINTR	filename (1)
MVS	POINT	dcbaddress , blockaddress (1-12) (2-12) (0)

Notes:

1. *VSE*: The address is that of a four-byte storage location containing the required record identification in the same form as it is obtained from the NOTE macro.
2. MVS: The blockaddrcss is the address of a fullword on a fullword boundary containing the required record identification in the same form as it is obtained from the NOTE macro.
3. The *MVS* POINT macro is valid only for BSAM and BPAM.
4. If you specify OPTCD=H in the data control block, the POINT macro instruction can perform record positioning on VSE tapes that contain embedded checkpoint records. Any embedded checkpoint records that are encountered during the record positioning are bypassed and are not counted as blocks spaced over. You must specify OPTCD=H in a JCL DD statement. Do not use the POINT macro instruction to backspace VSE 7-track tapes that are written in data convert mode and which contain embedded checkpoint records.

POINTS Macro

VSE	POINTS	filename (1)
MVS	POINT	dcbaddress , blockaddress (1-12) (2-12) (0)

The POINTS macro in VSE causes tapes to be rewound and positioned to the first record following the label set. To achieve this in MVS, you must specify the hexadecimal value 00000001 in the blockaddress field.

RELSE Macro

VSE	RELSE	filename (1)
MVS	RELSE	dcbaddress (1-12)

The function of the RELSE macro is the same under VSE and MVS. The MVS RELSE macro is valid only for QSAM and QISAM.

TRUNC Macro

VSE	TRUNC	filename (1)
MVS	TRUNC	dcbaddress (1-12)

The function of the TRUNC macro is the same under VSE and MVS. The MVS TRUNC macro is valid only for QSAM.

FEOV Macro

VSE	FEOV	filename (1)
MVS	FEOV	dcbaddress , REWIND (1-12) LEAVE

The basic functions of the VSE and MVS FEOV macros are the same. In MVS, volume positioning can be specified by the option operand; if no option is coded, the positioning specified in the OPEN macro is used. The MVS FEOV macro is valid for BSAM and QSAM.

GET / PUT Macros

VSE	GET PUT	filename , workname (1) (0)
MVS	GET PUT	dcbaddress , area address (1-12) (2-12) (0)

The functions of the VSE and MVS GET/PUT macros are the same.

Figure 35 on page 302 shows a comparison of the MVS DCB and VSE DTFMT macros. Figure 36 on page 303 shows an example of using some of the preceding macros in a program.

VSE DTFMT	MVS DCB DSORG=PS
BLKSIZE = nnnnn	BLKSIZE = nnnn
DEVADDR = SYSxxx	N/A
EOFADDR = xxxxxxxx	EODAD = xxxxxxxx
FILABL = xxxx	LABEL = (in DD statement)
IOAREA1 = xxxxxxxx	BUFNO = 1
or	
IOAREA1 = xxxxxxxx	BUFNO = 2 or more
IOAREA2 = xxxxxxxx	
ASCII = YES	OPTCD = Q
BUFOFF = nn	BUFOFF = (n)
ERREXT = YES	SYNAD = xxxxxxxx
ERROPT = IGNORE	EROPT = ACC
SKIP	SKP
	ABE
ERROPT = xxxxxxxx	SYNAD = xxxxxxxx
IOREG =(r)	MACRF= (..L..)
LABADDR = xxxxxxxx	EXLST = xxxxxxxx
(standard labels)	LABEL = (,SUL) (in DD statement)
NOTEPNT = YES	MACRF=(RP,WP)
POINTS	
READ = xxxxxxxx	OPEN Clacro option
RECFORM = xxxxxx	RECFM= xxx
RECSIZE = nnnn	LRECL = nnnn
= (r)	
REWIND = xxxxxx	OPEN macro option
SEPASMB = YES	User must code the DCB
TPMARK = NO	Standard in MVS
TYPEFLE = INPUT	MACRF = (G...)
OUTPUT	(P...)
	INPUT/OUTPUT are also specified in OPEN macro.
TYPEFLE =WORK	MACRF= (R...,W...)
VARBLD = (nn)	User must supply length of logical record +4 in LRECL field before issuing a PUT.
WLRERR = xxxxxxxx	SYNAD = xxxxxxxx
WORKA = YES	MACRF = (..M..)

Figure 35. Comparison of the DTFMT and DCB Macros

```
         OPEN  TAPE
VSE      PUT   TAPE
           .
         CLOSE TAPE
RECORD1  DS    2000C
TAPE     DTFMT DEVADDR=SYS005,TYPEFLE=OUTPUT,                          C
               FILABL=STD,IOAREA1=RECORD1,                           C
               HDRINFO=YES,IOREG=(5),                                C
               RECFORM=FIXBLK,BLKSIZE=2000,                          C
               RECSIZE=100,REWIND=NORWD
         MTMOD RECFORM=FIXBLK


         OPEN  (TAPE,(OUTPUT,LEAVE))
         LA    5,RECORD1
MVS      PUT   TAPE,(5)
           .
         CLOSE (TAPE,(LEAVE))
RECORD1  DS    CL100
TAPE     DCB   DDNAME=TAPEDD,DSORG=PS,MACRF=(PM),                      C
               RECFM=FB,BLKSIZE=2000,LRECL=100
```

Figure 36. Tape File Programs in VSE and MVS

13.2.6.8 LIOCS Device-independent File Definition

Under VSE, when using the DTFDI macro to define your file, your entire program should be device-independent. Under MVS, every program should also be device-independent for optimum use from the operating system. In revising a VSE program with a DTFDI to run under MVS:

- Code RECFM=F in the MVS DCB macro. DTFDI does not need the RECFORM parameter because only fixed, unblocked records are supported. If you omit RECFM in MVS, undefined is assumed.
- If the DTFDI specified DEVADDR=SYSLST or DEVADDR=SYSPCH, the MVS DCB must specify RECFM=FM because the first byte of the VSE output area contains a control character. For increased flexibility, supply the RECFM and DEVADDR parameters, and all other parameters describing data set characteristics in the DD statement instead of in the problem program.

Figure 37 on page 304 shows a comparison between the VSE DTFDI and the MVS DCB macros.

VSE DTFDI	MVS DCB DSORG=PS
DEVADDR = SYSxxx	DDname (in DD statement)
IOAREA1 = xxxxxxxx	BUFNO = 1
or	
IOAREA! = xxxxxxxx	BUFNO = 2 or more
IOAREA2 = xxxxxxxx	
EOFADDR = xxxxxxxx	EODAD = xxxxxxxx
ERROPT = xxxxxxxx	SYNAD = xxxxxxxx
ERROPI = IGNORE	EROPI = ACC
SKIP	SKP
	ABE
	MACRF =(G...)
	= (p...)
	RECFM = FA
	= FM
	BLKSIZE = nnnn
IOREG = (r)	MACRF = (..L..)
RECSIZE = nnn	LRECL = nn
SEPASMB = YES	User must code the DCB
WLERR = xxxxxxxx	SYNAD = xxxxxxxx

Figure 37. Comparison of DTFDI and DCB macros

13.2.6.9 LIOCS Console File Definition

The DTFCN has no equivalent in MVS because reading and writing via the console is not handled at the GET/PUT level. MVS provides the WTO macro for writing on the console and the WTOR macro for writing a message and reading a reply. Therefore, you must replace a PUT macro for the console by either a WTO macro to display a message or a WTOR macro for displaying a message and reading a reply.

Do not display messages on the console unless they are necessary. The system writes many messages to the operator, and extra messages could lead to confusion and hinder the performance of the installation.

MVS	WTO WTOR	'message' 'message', reply address, reply length, ecbaddress (2-12) (2-12) (2-12)

13.2.6.10 LIOCS Sequential File Definition on Direct Access Devices

Devices In VSE and MVS, sequential DASD files are processed in the same way.

OPEN Macro

VSE	OPEN(R)	filename ,... (r1)
MVS	OPEN	dcbaddress , option1, option2 ,... (2-12)

Options

	Option 1	Option 2
QSAM	INPUT	,REREAD
	OUTPUT	,LEAVE
	UPDAT	,DISP
	EXTEND	
	INPUT	
	EXTEND	
	OUTPUT	,REREAD
BSAM	INOUT	,DISP
	UPDAT	,LEAVE
	OUTIN	
	OUTINX	

Note: Any number of dcbaddresses and associated options may be specified in the OPEN macro instruction.

CLOSE Macro

VSE	CLOSE(R)	filename ,777 (r1)
MVS	CLOSE	dcbaddress option ,... ,TYPE=T (2-12)

1. Options

   REREAD
   LEAVE
   FREE
   DISP

2. If you omit the option, the following positioning occurs:

   If you code TYPE=T, LEAVE is assumed (BSAM only).
   If you don't code TYPE=T, DISP is assumed (BSAM only).

3. You can code CLOSE with TYPE=T to temporarily close sequential data sets on direct access volumes processed with BSAM. When you use TYPE=T, the DCB used to process the data set maintains its open status, and you don't have to issue another OPEN macro to continue processing the same data set. A request to temporarily close a data set causes MVS to process labels, modify some of the fields in the system control blocks for that data set, and reposition the volume (or current volume in the case of multivolume data sets) in much the same way that the normal CLOSE macro does. When you code TYPE=T, you can specify that the volume either be positioned at the end of data (the LEAVE option) or be repositioned at the beginning of data (the REREAD option). When a DCB is shared among multiple tasks, the task that opened the data set must also close it; however, a subtask of the task that opened the DCB can issue the CLOSE macro with the TYPE=T option.

GET / PUT Macros

VSE	GET PUT	filename ,workname (1) (0)

MVS	GET PUT	dcbaddress area address (2-12) (2-12) (1) (0)

CNTRL Macro

There is no equivalent for the VSE CNTRL macro. The MVS CNTRL does not support DASD devices.

RELSE Macro

VSE	RELSE	filename (1)
MVS	RELSE	dcbaddress (1-12)

The VSE and MVS functions of this macro are the same. RELSE causes the remaining records in a buffer to be ignored.

TRUNC Macro

VSE	TRUNC	filename (1)
MVS	TRUNC	dcbaddress (1-12)

The VSE and MVS functions of this macro are the same. TRUNC causes the next logical record to be written as the first record of the next block.

ERET Macro

The VSE ERET macro enables a problem program ERROPT or WLRERR routine to return to IOCS and specify an action to be taken. In MVS, this is approximated by the SYNAD routine, which is an optional error analysis routine that is given control when an uncorrectable I/O error occurs. The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system.

For BSAM, if control is returned to the system, the system returns control to the problem program and proceeds as though no error had occurred. If you omit the SYNAD operand, the task is abnormally terminated when an uncorrectable I/O error occurs.

For QSAM, if the error condition was the result of a data validity error, the control program takes the action specified in the EROPT operand; otherwise, the task is abnormally terminated. The control program takes these actions when you omit the SYNAD operand or when the 2.spr analysis routine returns control. Uncorrectable I/O errors resulting from channel operations or direct access operations that make the next record inaccessible cause the task to be abnormally terminated regardless of the action specified in the EROPT operand. The action specified by EROPT is one of these three:

ACC Specifies that the problem program accepts the block causing the error. This action can be specified when a data set is opened for INPUT, RDBACK, UPDAT, or OUTPUT (OUTPUT applies to printer data sets only).

SKP Specifies that the block that caused the error is skipped. Specifying SKP also causes the buffer associated with the data block to be released. This function can be specified when a data set is opened for INPUT, RDBACK, or UPDAT.

ABE Specifies that the error results in the abnormal termination of the task. This action can be specified when the data set is opened for INPUT, OUTPUT, RDBACK, or UPDAT.

If you omit the EROPT operand, the ABE action is assumed.

READ Macro

VSE	READ	filename ,SQ, area length (1) (0) (r1) S
MVS	READ	decbname,SF, dcbaddress , area address , (2-12) (2-12) length (2-12) 'S'

Notes:

1. If the OPEN macro specifies UPDAT, you must use the execute form of the READ macro.
2. You must test the input operation for completion by using the CHECK macro instruction.

WRITE Macro

VSE	WRITE	filename , UPDATE , area , length (1) SQ (0) (r)
MVS	WRITE	decbname,SF, dcbaddress area address length (2-12) , (2-12) , (2-12) 'S'

Notes:

1. If the OPEN macro specifies UPDAT, you must use the execute form of WRITE.
2. You must test the output operation for completion by using the CHECK macro.

CHECK Macro

VSE	CHECK	filename control-address (1) , (0)
MVS	CHECK	decbaddress ,DSORG= IS (1-12) ALL

Notes:

1. The decbaddress must be the same as used in the READ or WRITE macro (decbname).
2. If the I/O operation did not complete successfully, the error analysis routine (SYNAD) is given control if you have provided one.
3. The following conditions are also handled:
   - When the system is reading, volume switching is automatic. The end-of-data-set (EODAD) routine is given control if an input request is made after all the records have been retrieved.
   - When the system is writing, additional space on the device is obtained when the current space is filled and more WRITE macros have been issued.

POINTW / POINTR Macros

VSE	POINTW POINTR	filename , address (1) (0)
MVS	POINT	dcbaddress , blockaddress (2-12) (2-12) (1) (0)

Notes:

1. Blockaddress is the address of a fullword on a fullword boundary containing the required record identification in the same form as it is obtained from the NOTE macro.
2. If you use WRITE SQ after POINTR in VSE, set the 0 of TTR0 to 1 in MVS for the same result.

POINTS Macro

VSE	POINTS	filename (1)
MVS	POINT	dcbaddress , blockaddress (2-12) (2-12) (1) (0)

Notes:

1. The POINTS macro in VSE causes repositioning of the file to the lower limit of its first extent.
2. The POINT macro in MVS positions the data set to the block indicated in the blockaddress field, which contains TTRz where:
   - TT is the relative track number.
   - R is the block number on that track.
   - z is zero or one, if it is one, the block following the TTR block is referenced.

   You can specify the first block of a direct access device data set by either hexadecimal 00000001 or 00000100.
3. The MVS POINT macro is valid only for BSAM and BPAM.

NOTE Macro

VSE	NOTE	filename (1)
MVS	NOTE	dcbaddress (1-12)

The MVS NOTE macro is valid only for BSAM and BPAM.

NOTE returns the following information:

VSE-register 0: 00nn
VSE-register 1: CCHR
MVS-register 1: TTR0

where

```
nn = Unused space remaining on the track following the end of the
     identified record.
C  = Cylinder number.
H  = Track number.
T  = Relative track number.
R  = Block number of that track.
```

FEOVD Macro

VSE	FEOVD	filename (1)
MVS	FEOV	dcbaddress (1-12)

The functions of the VSE FEOVD and MVS FEOV macros are the same.

Figure 38 on page 310 shows a comparison of the operands of the DTFSD and DCB macros. Figure 39 on page 311 shows a sample of the I/O macros used with sequential direct access files in VSE and MVS.

VSE DTFSD	MVS DCB DSORG=PS
BLKSIZE = nnnn	BLKSIZE = nnnn
EOFADDR = xxxxxxxx	EODAD = xxxxxxxx
DELETFL = NO	DISP = (in DD statement)
DEVADDR = SYSxxx	N/A
DEVICE = nnnn	UNIT = (in DD statement)
ERROPT = IGNORE	EROPT = ACC
SKIP	SKP (QSAM only) (in DD stmt)
	ABE
ERROPT = xxxxxxxxx	SYNAD = xxxxxxxx
ERREXT = YES	SYNAD = xxxxxxxx
FEOVD = YES	Not required
HOLD = YES	This function can be implemented using the ENQ/DEQ logic of MVS for a specific resource. DISP=SHR in DD statement.
IOAREA1 = xxxxxxxx	BUFNO = 1
or	
IOAREA1 = xxxxxxxx	BUFNO = 2 or more
IOAREA2 = xxxxxxxx	
IOREG = (r)	MACRF = (..L..)
LABADDR = xxxxxxxx	EXLST = xxxxxxxx
RECFORM= xxxxxx	RECFM = xxx
RECSIZE = nnnn	LRECL = nnnn
(r)	
SEPASMB = YES	User must code the DCB
TRUNCS = YES	MVS assumes truncated blocks unless RECFM=(...S) is specified.
TYPEFLE = INPUT	MACRF = (G...)
OUTPUT	(P...)
	INPUT/OUTPUT are also specified in OPEN
TYPEFLE = WORK	MACRF = (R...,W...)
UPDATE = YES (for INPUT	MACRF = (R...,W...)
files only)	UPDAT is specified in OPEN macro
UPDATE = YES (for WORK	MACRF = (R...,W...)
files only)	
VARBLD = (r)	Not required
VERIFY = YES	OPTCD = W
WLRERR= xxxxxxxx	SYNAD= xxxxxxxx
WORKA= YES	MACRF= (..M..)

Figure 38. Comparison of the DTFSD and DCB Macros

```
         OPEN  SAMFILE
           .
         WRITE SAMFILE,SQ,DATA
VSE      CHECK SAM FILE
                .
         CLOSE SAMFILE
SAMFILE  DTFSD DEVADDR=SYS005,DELETFL=NO,                              C
               DEVICE=3340,RECFORM=FIXUNB,                             C
               BLKSIZE=80,TYPEFLE=WORK,VERIFY=YES
         SDMODW


         OPEN  SAMFILE,(OUTPUT))
                 .
         WRITE DECB,SF,SAMFILE,DATA
MVS      CHECK DECB
                 .
         CLOSE SAMFILE
SAMFILE  DCB   DDNAME=SAMDD,RECFM=F,DSORG=PS,                          C
               BLKSIZE=80,MACRF=(W)
```

Figure 39. Sequential DASD FILE Program in VSE and MVS

13.2.6.11 LIOCS Direct Access File Definition

The following text discusses modifying DASD direct access files for use under MVS.

General Considerations

File definition is accomplished under VSE by a DTFDA macro and by a DCB macro under MVS. Equivalents of certain DTFDA parameters are not specified in the DCB. You must specify some of these in a DD statement or in the imperative macro instructions. Figure 40 on page 312 shows a comparison of the operands of the DTFDA and DCB macros.

VSE DTFDA	MVS DCB DSORG=DA
BLKSIZE = nnnn	BLKSIZE = nnnn
DEVICE = nnnn	UNIT = (in DD statement)
ERRBYTE = xxxxxxxx	(See description of SYNAD routine)
IOAREA1 = xxxxxxxx	Area address (READ/WRITE macro)
SEEKADR = xxxxxxxx	Not required (READ/WRITE macro)
TYPEFLE = xxxxxx	OPEN macro option
AFTER = YES	MACRF = (..WA..)
DEVADDR = SYSnnn	UNIT = (in DD statement)
ERREXT = YES	SYNAD = xxxxxxxx
HOLD = YES	This function can be implemented by using the ENQ/DEQ logic of MVS for a specific resource or by requesting exclusive control of a data block thru MACRF = (..X..)
IDLOC = xxxxxxxx	(See description of updating the file)
KEYARG = xxxxxx	Keyaddress (READ/WRITE macro)
KEYLEN = nnn	KEYLEN = nnn
LABADDR = xxxxxxxx	EXLST = xxxxxxxx
READID = YES	MACRF = (..RI..)
READKEY = YES	MACRF = (..RK..)
WRITEID= YES	MACRF = (..WI..)
WRITEKY = YES	MACRF = (..WK..)
RECFORM= xxxxxx	RECFM = xxx
RECSIZE = (nn)	Length (READ/WRITE macro) or DCB LRECL = nnnnn
RELTYPE = xxx	OPTCD = (...,R...).No equivalent for DEC
SEPASMB = YES	User must code the DCB
SRCHM = YES	LIMCT= n,OPTCD = E
TRLBL = YES	EXLST = xxxxxxxx
VERIFY = YES	OPTCD = W
XTNTXIT = xxxxxxxx	EXLST= xxxxxxxx

Figure 40. Comparison of DTFDA and DCB Macros

Error Bytes

The error bytes used in VSE to test for successful completion of an I/O operation are equivalent to a two-byte exception code in MVS. The two-byte exception code is placed in the DECB (DECB+1) of the corresponding READ/WRITE macro after a WAIT or CHECK macro has been issued.

If you issue a WAIT macro (WAITF in VSE), test for successful I/O completion provide the appropriate actions.

In MVS, if you issue the CHECK macro, the system performs the test. If an unusual condition occurs, the system branches to your SYSNAD routine if you have provided one, or it abnormally terminates the job step.

The information provided in the two bytes is different in VSE and MVS. Therefore change any parts of your programs that refer to these error bytes. Figure 41 on page 313 gives a comparison of the VSE and MVS exception codes.

ERROR	VSE		MVS	
	Byte	Bit	Byte	Bit
Wrong-length record	0	1	2	1
Nondata transfer error	0	2	2	6
Space not found on track	0	4	2	2
Reference outside extents of data set or file	0	7	3	3
Data check in count area	1	0	2	4
Track overrun	1	1	**	**
End-of-cylinder	1	2	**	**
Data check when reading key or data	1	3	2	4
Record not found	1	4	2	B
End-of-file	1	5	2	5
End-of-volume	1	6		
Invalid request	*		2	3
Uncorrectable error other that I/O	*		2	6
Read with exclusive control not preceded by write with exclusive control	*		2	7
WRITE macro used when DCB specified input	*		3	1
Extended search specified with DCB IIMCT = 8	*		3	2
WRITE Mith ID addressed RO	*		3	4
Key was specified as search argument when KEYLEM = 8 or no key address was given	*		3	5
Request for options not in the DCB	*		3	6
Attempt to add fixed-length record with key beginning with hex 'FF'.	*		3 7	

* These errors are not included in the VSE exception codes.

** These conditions do not occur under MVS.

Figure 41. VSE Error Bytes and MVS Exception Code Bits

READ Macro

VSE	READ	filename (1) , KEY ID
MVS	READ	ecbname,type, R / RU , dcbaddress (2-12) , area address (2-12) , length (2-12) 'S' keyaddress (2-12) , blockaddress (2-12) , next address (2-12) 'S' 0

Notes:

1. TYPE - DI, DIF, DIX, DK, DKF, or DKX. R or RU can be suffixed to the type code only if spanned records are being processed. R signifies that the system is to return the relative track address of the next data record in the area specified by the next address operand. RU signifies that the system is to return the relative track address of either the next capacity record (R0) or data record, whichever occurs first. If R or RU is used, you must code the length operand as 'S'.
2. 'S' - system will supply the operand if you specify 'S'.

WRITE Macro

VSE	WRITE	filename (1) , KEY / ID / RZERO / AFTER / AFTER,EOF
MVS	WRITE	decbname,type, dcbaddress (2-12) , area address, (2-12) ′S′ / length (2-12) ′S′ , key address (2-12) ′S′ 0 , block address (2-12)

Notes:

1. TYPE = DA, DAF, DI, DIF, DIX, DK, DKF, or DKX.
2. ′S′ - system supplies the operand if you specify ′S′.
3. If the key is not written or used as a search argument, zero is specified instead of keyaddress.

CNTRL Macro

There is no equivalent in MVS BDAM to the VSE CNTRL macro. The MVS CNTRL macro does not support DASD devices.

WAITF, OPEN and CLOSE Macros

VSE	WAITF	filename (r1)
MVS	WAIT / CHECK	number of events, ECB = address / ECBLIST = address / dcbaddress (1-12) ,DSORG = IS / ALL

VSE	OPEN(R)	filename (r1) ,...
MVS	OPEN	dcbaddress (2-12) , INPUT / OUTPUT / UPDAT

VSE	CLOSE(R)	filename (r1l)
MVS	CLOSE	dcbaddress (2-12) , FREE / DISP ,...

Notes:

1. DISP is assumed when no positioning option is specified in the CLOSE macro. The volume is then positioned according to the position implied by the DISP parameter of the DD statement.
2. Any number of dcbaddresses may be specified in the OPEN and CLOSE macros.

13.2.6.12 Track and Record Addressing

Track Addressing

In VSE and MVS, you can make track references by using either the actual or relative addressing technique. The track reference field for actual addressing in VSE and MVS is of the form MBBCCHHR. The contents of the M byte is different in VSE and MVS. In VSE, M represents volume number, starting with zero for the first volume. This must be increased by one for each subsequent volume.

In the MVS data set control block (label), M contains the extent sequence number, which always starts with zero, on each volume. This must be increased only for extents within the same volume. In the MVS data extent block, M is a pointer to extent information. Therefore, if actual addressing is required, you must change your calculation routine to meet the MVS requirements. However, you should use relative addressing instead of actual addressing because of the simplicity in calculating the proper track number.

In the case of relative track addressing, the correct disk addresses are generated by the control program. If you use actual addressing, you must check the extents on that volume to make sure nothing else is written there. Actual addressing does not allow you to take advantage of some of the MVS facilities and may impair the performance of the system. Relative addressing allows the control program to place the data set where it is most convenient. It does all the necessary checking of extents. With relative addressing, the system keeps track of each data set, thus making programming easier and system use more efficient.

Record Addressing

Within a track, records may be addressed either by their record number (ID) or by key.

Record Addressing by ID

Provide the record number in the R byte of the track reference field.

VSE	READ	filename,ID
MVS	READ	ecbname,DI,...,blockaddress

Blockaddress points to a field containing the complete identification of the record.

Record Addressing by KEY

Supply the key as follows:

VSE	READ	filename,KEY KEYARG and KEYLEN operands are required in the DTFDA macro.
MVS	READ	decbname, DK,...,keyaddress,blockaddress

Keyaddress points to a field containing the key of the record for which you are searching. Blockaddress points to a field containing sufficient information to identify the track on which the search is to begin.

Reference Methods

The following paragraphs describe record reference by ID and record reference by key. In each method, relative track addressing and actual physical addressing are applicable to VSE and MVS; relative block addressing is applicable only to MVS. The relative block addressing technique locates a block by its position relative to the first block of the data set.

Record Reference by ID

Under VSE, records can be referenced by ID when you specify READID and/or WRITEID in the DTFDA. You must also supply both the track information and the record number in the field specified by SEEKADDR.

Under MVS, records can be referenced by ID when you specify MACRF=(...I...) in the DCB and the type parameter of the READ/WRITE macro contains I (see Figure 42 on page 317).

Reference Method	VSE	MVS
Relative Track Addressing	Assumed if DSKXTNT is specified. RELTYPE=HEX (the default) requires the hexadecimal form TTTR. RELTYPE=DEC requires the zoned decimal for TTTTTTTTRR. In both cases, the R byte(s) must contain the actual record number of the record on the track.	Assumed if OPTCD does not contain R or A. The field pointed to by blockaddress must contain TTR. There is no equivalent to RELTYPE =DEC in MVS. The form must be converted to hexadecimal.
Relative Block Addressing	No equivalent.	Assumed if MACRF contains I and OPTCD contains R. The field pointed to by blockaddress must contain BBB (binary). The address of the first record is 000.
Actual Physical Addressing	Assumed if DSKXTNT is not specified. The address must be in the form MBBCCHHR. The R is ignored.	Assumed if OPTCD contains an A. The field specified by blockaddress must contain MBBCCHHR. The M byte is different in VSE and MVS. See the description of the M byte under Track and Record Addressing.

Figure 42. Record Reference by ID in VSE and MVS

Record Reference by KEY

Under VSE, records can be referenced by key when you specify READKEY and/or WRITEKEY in the DTFDA. Before the WRITE macro is executed, the field specified by KEYARG in the DTFDA must contain the key of the record for which you search. The field specified by SEEKADDR must contain the track address. If you specify KEYARG, a record with that must already exist on the track or else a no record found condition occurs.

Under MVS, records can be referenced by key when you specify MACRF=(...K...) in the DCB and the type parameter of the READ,WRITE macro contains K. The field specified by keyaddress in the READ/WRITE macro must contain the actual key of the record for which you search. Blockaddress contains the search start address (see Figure 43 on page 318).

Reference Method:	VSE	MVS
Relative Track Addressing	Assumed if DSKXTNT is specified. RELTYPE=HEX (the default) requires the hexadecimal form TTTR. RELTYPE=DEC requires the zoned decimal form TTITTTTTRR. RR must be 00.	Assumed if OPTCD does not contain A. The field specified by blockaddress must contain TT in binary. There is no equivalent to RELTYPE=DEC in MVS. The form must be converted to hexadecimal.
Relative Block Addressing	No equivalent.	OPTCD=R must be specified in the DCB parameter. The field specified by block-address must contain BB in binary. The address of the first record is 000.
Actual Physical Addressing	Assumed if DSKXTNT is not specified. The address must be in the form MBBCCHHR. The R must be 0. The field specified by KEYARG must contain the key of the record.	Assumed if OPTCD contains an A. The field specified by blockaddress must contain MBBCCHHR. The M byte is different in VSE and MVS. See the description of the M byte under Track and Record Addressing.

Figure 43. Record Reference by KEY in VSE and MVS

Direct Access File Processing

In VSE, parameters required for creating or processing a DAM file are supplied or made known through the DTFDA macro instruction operands and using the READ/WRITE macros. For MVS, this information is in the DSCB, DCB, and READ/WRITE macros. Some information is also supplied in optional fields of the DD statement.

Figure 44, Figure 45 on page 319, Figure 46 on page 319, Figure 47 on page 320, Figure 51 on page 325, Figure 48 on page 320, Figure 49 on page 321 and Figure 50 on page 324 show parts of VSE programs with their MVS counterparts for loading and processing DAM files.

```
         OPEN     (DAMFILE,(UPDAT))
           .
           .
         READ     DECBUPOT,DI,,,,OLDKEY,,MF=E
         CHECK    DECBUPDT
           .
           .
         READ     DECBUPDT,DI,DAMFILE,'S','S',,                        C
                  BLOCKADDR,MF=L
           .
           .
DAMFILE  DCB      ...DSORG=DA,MACRF=(RISC,WIC),                        C
                  ...OPTCD=R,BUFL=58...
           .
           .
```

Figure 44. Updating a DAM File under MVS

```
         OPEN      (DAMFILE,(OUTPUT))
           .
           .
         WRITE     DECBADD,DI,DAMFILE,DATA,'S',KEY,BLOCKADDRESS
         CHECK     DECBADD
           .
           .
DAMFILE  DCB       ..MACRF=(WICS),DSORG=DA,OPTCD=R...
           .
           .
```

Figure 45. Adding to a DAM File under MVS

Loading a DAM File (Fixed-Length Records with keys)

Figure 46 and Figure 47 on page 320 illustrate an example of sequentially loading a DAM file consisting of fixed-length records with keys. A direct addressing technique is used, which provides unique correspondence between each key and its relative or actual position on the disk (VSE) or within the data set (MVS), similar to a sequential data set. The input file resides on tape and is a set of records sorted into ascending order by keys. Each record consists of 50 bytes (a three-byte key field followed by 47 bytes of data).

```
         OPEN  DAMFILE,TAPE
WRITERO  WRITE DAMFILE,RZERO
         WAITF DAFMFILE
           .
GET      GET   TAPE
           .
         WRITE DAMFILE,AFTER
         WAITF DAMFILE
           .
EOF      WRITE DAMFILE,AFTER,EOF
         WAITF DAMFILE
         CLOSE DAMFILE,TAPE
           .
DAMFILE  DTFDA BLKSIZE=58,ERRBYTE=ERROR,                              C
               IOAREA1=OUTPUT,SEEKADDR=ADDR,                          C
               TYPEFLE=OUTPUT,AFTER=YES,                              C
               DSKXTNT=3,KEYLEN=3,RELTYPE=HEX,                        C
               VERIFY=YES,DEVICE=3340
         DAMOD AFTER=YES,ERREXT=YES,RELTRK=YES
           .
TAPE     DTFMT .....
         MTMOD
```

Figure 46. Loading a Sequential DAM File under VSE

The VSE program writes the capacity record (R0) and uses WRITE AFTER to sequentially write each record, so that the capacity record is checked for available space on that track before each record is written. In VSE, the total amount of space allocated to the file is indicated by the extents allocated to it, but it can be extended in a later run.

```
         OPEN  (DAMFILE,(OUTPUT),TAPE,(INPUT))
GET      GET   TAPE
                 .
         WRITE DECB1,SF,DAMFILE,(10)
         CHECK DECB1
                 .
         WRITE DECB2,SD,DAMFILE,DUMMYREC
         CHECK DECB2
                 .
TAPE     DCB   .......
                 .
DAMFILE  DCB   DDNAME=OSDAMDD,DSORG=PS,                              C
               MACRF=(WL),SYNAD=DATRTST,                             C
               RECFM=F,KEYLEN=3,BLKSIZE=47
```

Note: DSORG=PS must be specified in the DCB. However, DSORG=DA must be specified in the DD statement.

Figure 47. Loading a Sequential DAM File under MVS

Under MVS, you can create a like data set by using the WRITE SF macro. You can also use the WRITE SD macro to fill any tracks remaining to the end of the data set with dummy records (if desired for future additions).

Figure 48 and Figure 49 on page 321 illustrate an example of loading a preformatted direct access file by record ID.

```
         OPEN  (DAMFILE,(UPDAT),TAPE,(INPUT))
GETTAPE  GET   TAPE,INPUT
                 .
         WRITE DECBADD,DA,DAMFILE,DATA,'S',KEY,                      C
               BLOCKADDR
         CHECK DECBAOD
EOF      CLOSE (DAMFILE,, TAPE)
                 .
TAPE     DCB   ...
DAMFILE  DCB   DDNAME=OSDAMDD,DSORG=DA,                              C
               MACRF=(WAC),OPTCD=E,LIMCT=5,                          C
               SYNAD=DATRTST
```

W WRITE macro is used.
A Records are to be added.
C CHECK macro is used.

Figure 48. Loading a Random DAM File under MVS

```
                                                    Notes
          .
          OPEN  (ROFILE,(OUTPUT),TAPE)
WRITERO   WRITE DECBRO,SZ,ROFILE
          STC   15,RC                               (1)
          CHECK DECBRO
          CLI   RC,X'00'
          BE    WRITERO                             (2)
OPENDAM   OPEN  (DAMFILE,(OUTPUT))
          CLI   WC,X'10'                            (3,8)
          BE    WRITE
GET       GET   TAPE,WORK
          PACK  PKEY(2),KEY
          CVB   9,CVBKEY
          SR    8,8                                 (4)
          D     8,=F'37'
          LTR   8,8
          BNZ   *+10
          BCTR  9,0
          IC    8,=C'37'
          STH   9,TT
          STC   8,R
WRITE     WRITE DECBLOAD,DAF,DAMFILE,DATA,47,KEY,TTR (5)
          WAIT  ECB=DECBLOAD
          MVC   WC,DECBLOAD+2
          TM    WC,X'10'                            (6)
          BO    CLOSEDAM                            (7)
          CHECK DECBLOAD
          B     GET
CLOSEDA   AH    7,COUNT
          LA    7,0(7)
          STH   7,COUNT
          CH    7,THREE
          BH    BYPASS
          CLOSE (DAMFILE)
          B     WRITERO
BYPASS    NOTE  RECORD
          B     GET
EOF       CLOSE (DAMFILE)
          CLOSE (ROFILE,,TAPE)
          .
WORK      DS    0CL50
KEY       DS    CL3
DATA      DS    CL47
          DS    CL30
CVBKEY    DC    D'0'
PKEY      EQU   *-2
TTR       DS    0F
TT        DS    CL2
R         DS    CL1
          DS    CL5
*
```

Figure 49 (Part 1 of 3). Loading a DAM File of U. or V. Length Records under MVS

```
                                                  Notes
RC       DS    CL1
WC       DC    CLI'0'
COUNT    DC    H'0'
THREE    DC    H'3'
*

ROFILE   DCB   DDNAME=RODD,                    (9)                    C
               DSORG=PS,                                             C
               MACRF=(WL),                                           C
               SYNAD=OSDAERR1,                                       C
               BLKSIZE=47,                                           C
               KEYLEN=3,                                             C
               RECFM=U
*
DAMFILE  DCB   DDNAME=DAMDD,                  (9)                    C
               DSORG=DA,                                             C
               MACRF=(WAC),                                          C
               OPTCD=F,                                              C
               SYNAD=OSDAERR2
*
TAPE     DCB   DDNAME=TAPEDD,                                        C
               DSORG=PS,                                             C
               MACRF=(GM),                                           C
               EODAD=EOF
*
//GO.TAPEDD DD DSN=OSSAMFIL,
//             UNIT=3420,VOL=SER=NONE,
//             LABEL=(5,NL),
//             DISP=OLD,                                -
//             DCB=(BLKSIZE=50,RECFM=F)
//GO.RODD   DD DSN=UDAM,                        (9)
//             UNIT=3340,VOL=SER=WORK12,
//             SPACE=(47,(1000,100)),
//             DCB=(DSORG=DA),
//             DISP=(,KEEP)
//GO.DAMDD  DD DSN=UDAM,                        (9)
//             UNIT=3340,VOL=SER=WORK12,
//             DISP=OLD
```

Notes:

1. One track, in sequential order, is erased and a capacity record is written. On return, register 15 contains a return code which is zero if another available within the initial extents, and is 8 if the next track will require secondary space allocation.
2. If the return code was 0, a new track is initialized, otherwise the second DCB is opened and actual loading may commence for the area just cleared.
3. Initially this byte is set to zero and no branch will occur.

Figure 49 (Part 2 of 3). Loading a DAM File of U. or V. Length Records under MVS

4. Since 37 records (blocks) will fit on one 3340 track, the three-byte key (ranging from 001-999) is divided by 37 to get the relative track number and the remainder will be the number of the record on this track. If a remainder of O is computed, this record will still fit on the previous track; therefore TT = n, R = 0 will be reset to TT = n-1, R = 37 since a remainder of 37 may not be evaluated otherwise. A key may not consist of all zeros in this case, as a TT of -1 will result. It is, in fact, not necessary to maintain the R-byte, since the system does not use it if WRITE DA is specified.
5. A block is added to the data set. Type DA must be used and the data length must be stated since RECFM=U was assumed.
6. A second error byte in the DECB is saved (which might not be necessary) and inspected for the ″Out of Extents″ condition.
7. If the condition was satisfied, the load DCB must be temporarily closed. Control is then given to the WRITER0 routine which requests secondary space allocation and, after getting the new extent, writes the capacity records for this area.
8. If any extents except the initial one(s) have been cleared, loading may resume after reopening the load DCB, however, the WRITE macro (5) has to be successfully executed before the load loop is reentered.
9. Note the different DD names and status of the data sets.

Figure 49 (Part 3 of 3). Loading a DAM File of U. or V. Length Records under MVS

In both figures, the file being loaded consists of 50-byte records. Each record has a three-byte key field followed by 47 bytes of data. The converted key field becomes the relative track address. Under VSE, the record is written in the first available space within the file extents on that track. Under MVS you must code the type field in the WRITE macro as DA. Then, each record you write replaces the first dummy record found (indicated in the key field) on the specified track. If the track is already filled but you have requested an extended search (DCB parameters LIMCT and OPTCD), the search is continued.

Loading a DAM File (Fixed-Length Records without keys)

Loading a file of fixed-length records without keys is similar to loading one when the records have keys, except that you cannot use the MVS WRITE SD) macro to write dummy records. You must define what a dummy record looks like when you create the file.

The DA type WRITE is not applicable in this case. You must retrieve a record, test it for indications of a dummy record, and after updating, rewrite the record.

Loading a DAM File (Undefined or Variable-Length Records)

For undefined or variable-length records, relative track addressing using keys is the best method. Before using any track when creating the file, write a capacity record (R0) and erase the rest of the track by issuing an SZ type WRITE instruction. If the total space needed for the data set is initially allocated, you must first write capacity records for all tracks. Then loading the data records consists of making additions to the initially empty file by using the MVS WRITE DA macro (similar to the VSE WRITE AFTER macro).

To create a file using this method under MVS, you would normally initialize each track by writing a capacity record (R0) and erasing the 2.sp of the track. In VSE, you would do this by using the WRITE RZERO macro; in MVS you use the WRITE SZ macro. However, in MVS, you need not update the track address because this is done automatically by the WRITE SZ macro. By testing register 15 for a non-zero value after each WRITE, you can determine when MVS has initialized all the tracks. Also, you need a second sequential DCB (DSORG=PS) for the WRITE SZ macro. An example of this procedure is shown in Figure 49 on page 321. The example also shows how secondary space allocation can be obtained if an out-of-extent condition occurs while you are creating the data set.

Processing a DAM File under VSE

Figure 50 illustrates how a DA file that has been loaded sequentially under VSE may be processed. Records are retrieved for updating purposes by key and the relative track number. When the record-not-found condition occurs, the transaction record whose key was used for the search is added to the disk file by a WRITE AFTER.

```
         OPEN      DAMFILE
           .
         READ      DAMFILE,KEY
         WAITF     DAMFILE
           .
           .
         WRITE     DAMFILE,KEY
         WAITF     DAMFILE
           .
           .
ADDITION WRITE     DAMFILE,AFTER
         WAITF     DAMFILE
           .
         CLOSE     DAMFILE
           .
DAMFILE  DTFDA     BLKSIZE=58,ERRBYTE=ERROR,                          C
                   IOAREA1=OUTPUT,SEEKADR=ADDR,                       C
                   TYPEFLE=INPUT,AFTER=YES,DSKXTNT=3,                 C
                   KEYARG=KEY,KEYLEN=3,VERIFY=YES,                    C
                   READKEY=YES,RELTYPE=HEX,                           C
                   WRITEKY=YES,DEVICE=3340
         DAMOD     AFTER=YES,ERREXT=YES,RELTRK=YES
```

Figure 50. Processing a DAM file under VSE

To process a randomly loaded file, use a similar process, but use READ ID for retrieving records and WRITE ID for updating and adding records.

Processing a DAM File under MVS

The procedure for adding records to a BDAM data set is similar to the one illustrated in Figure 50. The computation of the block address field varies according to the reference method used. For example, if the data set had been created sequentially, as in Figure 47 on page 320, record reference by block address only can be used. In this case, the coding might be as illustrated in Figure 42 on page 317.

When updating records, it is convenient to use the list and execute forms of READ/WRITE macros rather than the standard forms and to request dynamic buffering. To update a direct access file that was created sequentially, as in Figure 51 on page 325, use coding similar to that of the example in Figure 44 on page 318, which uses the list and execute forms of the READ and WRITE macros. The key does not need to be rewritten after a record has been retrieved and updated, so the key address field is specified as zero. You must specify BUFL because dynamic buffering was requested.

```
                .
                .
         OPEN  DAMFILE,TAPE
GETTAPE  GET   TAPE
                .
                .
         WRITE DAMFILE,ID
         WAITF DAMFILE
                .
                .
         CLOSE DAMFILE,TAPE


TAPE     DTFMT
         MTMOD
DAMFILE  DTFDA BLKSIZE=50,DEVICE=3340,DSKXTNT=3,                      C
               ERRBYTE=ERROR,IOAREA1=OUTPUT,                          C
               SEEKADDR=ADDR,TYPEFLE=OUTPUT,                          C
               RELTYPE=HEX,VERIFY=YES,WRITEID=YES
         DAMOD ERREXT=YES,RELTRK=YES
```

Figure 51. Loading a Random (Preformatted) DAM File under VSE

Multiple Search / Feedback

Under VSE, if you have specified SRCHM=YES, READ or WRITE KEY and IDLOC=xxxxxxxx, the system returns to you the address of the record currently being read or written. This address is placed in the field specified by IDLOC. VSE supplies the ID in the same form used in the SEEKADR location, except when physical IDs are involved. In that case, only the last five bytes of the physical ID, cchhr, are supplied instead of the complete relative ID including zeros.

If you use READ or WRITE ID (or READ or WRITE KEY without SRCHM), VSE returns the address of the next record. This address is placed in the field specified by IDLOC. When using relative addressing with IDLOC, all your extents, except the last extent for each file, should end on cylinder boundaries.

Under MVS, if the data set was created using relative track, key,. and extended search, you may request feedback to increase the speed of an update run. Feedback means that after the execution of a READ/WRITE macro, the address of the retrieved record is supplied by the system in the field specified by blockaddress.

Thus, if a READ requires an extended search over several tracks to locate a specific record by its key, the subsequent update WRITE may use the address supplied by the feedback option rather than repeat the entire search.

In MVS, to make a request for feedback, insert the letter F in the type code of a READ/WRITE macro (DIF, DAF and so on). The format of the blockaddress field after feedback, however, is determined by the OPTCD parameter. If the OPTCD parameter does not contain an F, feedback will be in the form of MBBCCHHR. If you code an F in the OPTCD parameter, the format of the feedback depends on the reference method used. Figure 52 provides details on reference methods and feedback formats.

Type or Reference					
Character-istics		Relative block number within the data set. You supply key	Relative track number and block identifi-cation.	Relative track number. You supply key in key field.	Actual device address.
		BBB	TTR	TT	MBBCCHHR
Feedback Format	Feedback Format	Binary value left justified in block address field.	TT=Track number (2-byte binary value) R=Record on track (1-byte binary value)	Relative track number (2-byte binary value)	
Minimum length (bytes) of Block-address field	3	3	2	8	
OPTCD = F	Relative Address Bytes	BBB 3	BBB 3	TTR 8	TTR 3
OPTCD NOT = F	Actual Address Bytes	MBBCCHHR 8	MBBCCHHR 8	MBBCCHHR 8	

Figure 52. MVS Feedback Formats

13.2.6.13 LIOCS Indexed Sequential Definition

Indexed sequential (ISAM) files should be converted to VSAM; that is, **ISAM files should not be used in MVS.** VSE ISAM files should either be converted prior to the migration to MVS (recommended) or during the conversion process.

13.2.6.14 PIOCS

EXCP is often used in VSE in association with card files (for example, SYSIPT), print files (for example, SYSLST) or the operator console (for example, SYSLOG). In MVS, EXCP can not be used for spool files or to dialog with the operator. VSE programs that use EXCP should be converted to use standard access method such as QSAM or the WTO/WTOR macros.

If IBM-supplied access methods are used in MVS, provide a DCB, issue an OPEN macro for the data set, and then process it by appropriate I/O macros. It is the function of the access routines to:

- Provide for CCWs.
- Construct the IOB (input/output block).
- Construct the ECB (event control block).
- Issue an EXCP (execute channel program) macro.

Subsequently, the I/O supervisor schedules the request, issues the START I/O instruction, handles interrupts, posts results in the form of completion codes, and, if necessary, executes error recovery routines. Therefore, if the usual access methods do not apply to a specific problem, you must include in your program the coding necessary to provide for the preceding access method functions.

Under VSE, only one control block, the CCB, is needed and this control block may be built by a macro. Under MVS, however, you must allocate and partially fill the input/output control block and the DECB through the use of DCs.

Overview of Programming Elements

In VSE, certain job control statements, system macros, and control blocks are used. For each of these major programming elements (except SEOV), there is a corresponding major MVS element. (See Figure 47 on page 320)

CCB Macro

Figure 53 shows the relationship between the VSE CCB macro operands and their MVS equivalents:

CCB OPERAND	MVS EQUIVALENT
SYSnnn	DDNAME field of DCB macro. After the OPEN macro has been executed, the unit control block address for the actual device is in the DEB.
Command-list-name X′nnnn′ (transmission bytes)	CCW address field of IOB. See discussion on CCB Fields.
Sense Address	No corresponding element. The system normally provides the first two bytes in the IOB.

Figure 53. Relationship between CCB operands and MVS Equivalents

DTFPH Macro

Figure 54 shows the correspondence between the operands of the DTFPH macro and their MVS equivalents:

DTFPH OPERAND	MVS EQUIVALENT
TYPEFLE	OPEN macro
ASCII	OPTCD=Q
CCWADDR	Channel program address field of IOB
DEVICE	UNIT parameter of DD statement
DEVADDR	UNIT parameter of DD statement
LABADRR	EXLST parameter in DCB allows for label checking exits
HDRINFO	No corresponding element
MOUNTED	DD statement
XTNTXIT	DEB

Figure 54. Relationship between DTFPH Macro and MVS equivalents

You can also use the file name of the DTFPH file as the ddname of the corresponding MVS data set.

13.2.6.15 Comparison of Physical IOCS Elements

VSE Major Elements	MVS Major Elements
Define the file for physical IOCS (DTFPH) macro and control block	Data Control Block (DCB) macro and control block
TLBL statement	DD statement
DLBL statement	DD statement
EXTENT statement	DD statement
ASSGN statement	DD statement
OPEN macro	OPEN macro
XTNTXIT routine	Data Extent Block (DEB)
LABADDR routine	EXLST in Data Control Block (DCB)
Command Control Block (CCB) macro and control block	Data Control Block (DCB) Input/Output Block (IOB) Event Control Block (ECB) Data Extent Block (DEB)
EXCP macro	EXCP macro
WAIT macro	WAIT macro
CLOSE macro	CLOSE macro
FEOV macro	FEOV macro
SECTVAL macro	TRKCALC
SEOV macro	Not applicable
LBRET macro	RETURN macro

Figure 55. Comparison VSE and MVS Major Elements

Chapter 14. RPG II

14.1 Migration from VSE to OS/390

The aim has been to make it easy to convert programs running under VSE to run under OS/390, with the minimum of changes to the source code. In fact, the source programs will run unmodified, except for the following:

- The command level interfaces with CICS/VS and DL/I VSE are not supported by OS/390 RPG II. They must not be specified on the Header Specifications form, and calculations that refer to CICS/VS and DL/I must not be entered on the Calculation Specifications form.
- Combined files do not exist in OS/390 RPG II. Such files must not be specified on the File Description Specifications form.

As explained below, many entries that are necessary for an RPG II program to run under VSE are ignored by OS/390. The compiler will issue a warning message, but will still produce object code.

When using spooled print files (JES2/JES3) with overflow printing you must use line counter specification, because channel 12 is not sensed.

Note

As an application development tool, you should consider adopting other application language enabling facilities which tend to take better advantage of S/370 architecture and MVS/XA, MVS/ESA environments; for example, COBOL, PL/I, CSP, FORTRAN.

14.1.1 Device Information

OS/390 allows data management information to be provided at object time by means of DD statements. Parameters such as record and block size, label information, or definition of disk space, can thus be changed without affecting the source code. The entry of these parameters on the File Description Specifications form is optional; **JCL specifications are recommended.**

Also optional under OS/390 is the definition of physical and logical device; the only device types that must be specified are CONSOLE and SPECIAL.

14.1.2 Print Files

1. For print files controlled by a line counter, without block length specification, RECFM must be specified in the JCL. LRECL is the record length specified in the file description specification plus 1 (for the machine control character).
2. For print files controlled by a line counter with block length specification, the RECFM in the DCB is the file format specified in the file description specification with machine control characters; it cannot be overwritten by JCL.
3. For print files without line control specifications and with block length specified in file descriptions:

- With the device type PRINTER or with a blank entry for device type and symbolic device SYSLST (DOS/VS RPG II only)
  - LRECL is the record length specified in the file description specification plus 1 (for the machine control character)
  - The first position of each record contains the machine control character. The RECFM in the DCB is the file format specified in the file description with machine control characters. It cannot be overwritten by JCL.
- With a blank entry for device type or with a device type other than print or punch device, the RECFM in the DCB is the file format specified in the file description and cannot be overwritten by JCL. Such files can be printed using an IBM utility program, for example, IEBGENER.

4. For print files without line control specifications and without block length specified in the file descriptions:
   - LRECL is the record length specified in the file description plus 1 (for the machine control character)
   - RECFM and BLKSIZE must be specified in the JCL.

14.1.3 Tape Labels

Whereas in DOS/VS RPG II an exit may be entered on the File Description Specifications form for handling non-standard tape labels, under OS/390 such labels must be handled by a system-wide routine; the entry on the form is ignored. In DOS/VS RPG II, multivolume unlabeled tapes may be specified in the same way. With OS/390, this information is given in JCL statements.

14.1.4 Extent Exit

Exits for DAM files, to check whether the computed track address lies within the extents of the file, are unnecessary under OS/390. Such an extent exit will be ignored.

14.1.5 Processing Options

The options DECK/NODECK, LIST/NOLIST, TERM/NOTERM, and ERRS/NOERRS are conveyed to OS/390 by means of the PARM field in the EXEC statement of the JCL. The default values are DECK, LIST, ERRS, and NOTERM (see *OS/VS RPG II Installation Reference*, SC33-6122. The other permitted options, NOLINK, CATAL, and LINK, are realized by means of the procedures for compile, for compile and link, or for compile, link, and go.

14.1.6 File Access Methods

The file organizations are supported as in DOS/VS RPG II. The following list correlates the file type and the file processing.

Sequential files are supported as device-independent QSAM files, unless device-dependent features are used. This allows the user to decide at object time where his sequential data items are to reside.

Indexed sequential files are processed with QISAM or BISAM (or both). The OS/390 RPG II compiler chooses the appropriate file access method, according to the entries made on the File Description Specifications form.

Direct access method files are processed with BDAM.

VSAM files are handled in the same way as under VSE.

14.1.7 Calling COBOL Subprograms

In OS/390 RPG II, a CALL statement for a COBOL subprogram must not be preceded by a CALL ′ILBDSET0′.

14.1.8 Calling PL/I Subprograms

It is not possible to call PL/I subprograms from an OS/390 RPG II program.

Year 2000

For VSE see PTF UN95321 (APAR PN88472).

For OS/390 see PTF UN97731 (APAR PN90587).

Chapter 15. PL/I

The PL/I language compiler implemented on VSE is the DOS PL/I Optimizing Compiler (5736-PL3). In MVS, the PL/I language is implemented by the OS PL/I Optimizing Compiler Version 1 (5734-PL3), and OS PL/I Version 2 Optimizing Compiler (5668-910).

As the OS PL/I Version 2 Optimizing Compiler implements more of the PL/I language than Version 1 does, most source programs compiled on the VSE compiler can be compiled on either of the two MVS compilers with a minimum amount of change.

Note: The most current version of PL/I is PL/I for MVS & VM Version 1, Release 1. This compiler produces object code designed to run under Language Environment, a common run-time environment for several languages on this platform. Please refer to the *IBM PL/I for MVS & VM Compiler and Run-Time Migration Guide Release 1.1*, SC26-3118 for assistance with your migration to Language Environment.

For a comparison of VSE PL/I and MVS PL/I language elements, see the publication entitled *Developing Portable VSE Applications*, GC33-6367.

15.1 Functional Differences

15.1.1 EGCS (VSE) to DBCS (OS Version 2) Comments

The following information is provided to those PL/I users who are currently using DOS PL/I Optimizing Compiler′s Extended Graphic Character Set (EGCS) support. The Version 1 OS PL/I compiler also has the EGCS support. The Version 2 OS PL/I compiler made enhancements to EGCS support and renamed the support, Double Byte Character Set (DBCS).

EGCS support is limited to support for the GRAPHIC string constant and GRAPHIC data type. It introduced the GRAPHIC compiler option, which allows the installation (but not the user) to define the graphic control characters. These control characters consist of the shift-in, shift-out, graphic blank, graphic quote and graphic letter ′G′. The format of the graphic string constant (with a graphic ′G′ suffix) is fixed.

DBCS support provides for additional new function (including free-format input) over that provided with EGCS. For a comprehensive list of the DBCS support items, see *OS PL/I Version 2 Language Reference*, SC26-4308 and the *OS PL/I Version 2 Programming Guide*, SC26-4307.

VSE migrations to the OS PL/I Version 2 product, should not experience any code problems given that they didn′t modify EGCS code points; that is, upward compatibility is provided.

15.1.2 Extended Precision

Available with the MVS version of the PL/I compiler, extended precision floating point allows working with variables of two double-words. This extended precision is requested by specifying a precision greater than 16 for decimal float variables and 53 for binary float variables. DOS PL/I does not support extended precision floating point arithmetic.

15.1.3 Multitasking

Multitasking support in MVS introduces in PL/I a number of new statements. It is worth noting that these new functions are only invoked if the program calls them explicitly. For example, it is possible in a CALL macro to associate an EVENT-type control variable. It is then possible, at the end of the program, to test the termination of the associated task with a procedure invoked by a WAIT or with the function COMPLETION. Similarly, it is possible to modify with a single statement the relative priority of a sub-task:

```
PRIORITY(T1) = PRIORITY(TI) + 2;
```

These new functions are additions to the DOS version and do not affect the compatibility of the compilers.

15.1.4 Dynamic Loading of Dependent Programs

It is possible to dynamically load sub-programs written in PL/I separately from a main PL/I program. This is done by using the FETCH statement. This statement, as the RELEASE statement with which it is associated, is only available on MVS. FETCH causes loading of a LOAD module from LINKLIB or an execution library, while RELEASE frees the space occupied by the module. It is necessary to pass control to the sub-program by a CALL statement. Certain restrictions exist in its use due to the fact that there must not be any external references between the main program and the sub-program loaded dynamically. Particular attention must be paid to the use in the sub-program of files and controlled variables declared in the main program. Since FETCH and RELEASE statements do not occur in DOS PL/I programs, their presence in the MVS implementation is purely additive and has no impact on DOS-to-MVS conversion.

15.1.5 File Organization

The file organizations supported in DOS by the PL/I Optimizer are:

```
REGIONAL (1) and REGIONAL (3)
CONSECUTIVE
VSAM
INDEXED
```

In MVS, these are all supported and so are REGIONAL(2) and TCAM (TRANSIENT). A new feature in the use of REGIONAL or INDEXED files is the capability of requesting a lock at the record level to control simultaneous updating of the same file by several programs or tasks. This is the attribute EXCLUSIVE. If this attribute is applied to a PL/I file, then at the time of a READ, the record will be locked until a REWRITE or UNLOCK has been issued. It is equally possible to request the reading of a record without locking it: READ NOLOCK. This support uses the ENQ and DEQ facilities of MVS, and implies that all the programs sharing the file be written in PL/I such that the RNAME and QNAE generated are identical for the same record. This function is not available for VSAM.

15.1.6 Parameters Passed to a Main Program

It is possible to pass parameters to a PL/I program having the option MAIN by declaring the entry point as follows:

```
P:PROCEDURE(PARAM) OPTIONS(MAIN);
DCL PARAM CHAR(100) VARYING;
```

When control is passed to the program, the character string PARAM will contain the parameter passed from the PARM field of the EXEC statement. The length of the character string will be set to the number of characters passed to the program.

Note: A main PL/I program always expects to receive parameters for the PL/I environment (COUNT or REPORT or ISASIZE ....). It is therefore necessary to separate these parameters from those required by the program by the character '/'. For example:

```
EXEC PGM=MAINPLI;PARM='REPORT,ISA(60K)/RESTART'
```

The character string RESTART is allocated to the field PARAM and its length will be set to seven bytes. If the PL/I Checkout Compiler is used, this parm string may have three fields (Checkout Compiler translation parms (for example, SOURCE), PL/I interpret time parms, and user data).

15.1.7 %INCLUDE

It is possible to specify in a %INCLUDE macro the DDname of the library which is to be searched for the text to be copied. By default, PL/I will search the library defined by the DDname SYSLIB.

```
%INCLUDE DCLFIC; the library to be searched is defined by:
       //SYSLIB DD DSN=...
%INCLUDE MYLIB (DCLFIC); the library to be searched is defined by:
       //MYLIB DD DSN=...
```

In DOS PL/I, this syntax existed with single-letter library identifiers for DOS source sublibraries, (for example, %INCLUDE Q(memb);), so the user may have to supply DD cards for DDNAMES "P", "Q", and so on. It is much more efficient to include only from the default data set SYSLIB, since the MVS PL/I compilers will OPEN and CLOSE other "include" libraries at every reference.

15.2 Compiler Options

15.2.1 Options Specific to the DOS Compiler

15.2.1.1 CATALOG

This option is produced by the compiler from a CATALR statement. It is not valid in MVS PL/I. It is replaced by a control statement in the file specified by //SYSLIN DD DSN=.

15.2.1.2 DYNBUF

In MVS the buffers are always acquired dynamically by the compiler. This option is therefore suppressed.

15.2.1.3 LIMSCONV

An option of DOS PL/I to generate strong external references to PL/I conversion library modules only for those conversions deemed ″reasonable″ for the data types of variables that appear in GET DATA and GET LIST statements. Without LIMSCONV the whole PL/I conversion is of much less importance in virtual storage systems. MVS PL/I does not support it.

15.2.1.4 LINK

This option is replaced by the conditional execution function offered by MVS JCL. It therefore no longer exists in MVS.

15.2.1.5 NAME

This option is functionally equivalent in the two compilers. The control statement generated for the link edit has of course a different format. It allows, among other compiler possibilities, several independent PL/I programs to be compiled in a single pass and to link-edit these programs in one pass (batched compilations).

15.2.1.6 WORKFILE

This option was used to define the type of compiler workfiles, but has no use in MVS and is suppressed. (The access method modules being loaded dynamically at OPEN, the workfiles are independent of physical units.)

15.2.2 Options Specific to the MVS Compiler

15.2.2.1 GONUMBER

Analogous to the GOSTMT option, it gives the line number in the case of an abnormal termination, for example a conversion error, instead of the number of the instruction. It requires the NUMBER option. This option is used under TSO and CMS, the editors of these two time-sharing systems numbering the lines of the source program.

15.2.2.2 NUMBER

This option informs the compiler that the input is numbered.

15.2.2.3 SEQUENCE

This option indicates to the compiler where to find the line numbers in the source program.

15.2.2.4 STATEMENT

This option requests the compiler to number the instructions of the source program.

15.2.2.5 SMESSAGE or LMESSAGE

This option requests the compiler to supply messages in short or long format. This is particularly useful for interactive users using only slow terminals (printers).

15.2.2.6 IMPRECISE

This is used for 360/91 and 360/195 only.

15.2.2.7 INTERRUPT

This option requests that control be given to an ATTENTION type of ON- UNIT in case of attention at the terminal. It is only of interest for conversational programs written to execute under TSO or CMS, or when tuning a batch program written interactively under TSO or CMS when the PL/I CHECKOUT compiler is not available.

15.2.2.8 TERMINAL

This option is used only in an interactive environment. It allows the specification of additional options controlling the display of results at the terminal. The following options can be specified with the option TERM: AGGREGATE ATTRIBUTES ESD INSOURCE LIST MAP OPTIONS SOURCE STORAGE and XREF.

15.2.3 Execution Options

These are not unique to MVS!! For DOS, however, they apply only to CICS/VS transactions!

15.2.3.1 ISASIZE

This option allows control of the management of storage used by PL/I during the execution of the program. By default, PL/I obtains an ISA (Initial Storage Area) of half the available storage after the load module is loaded, unless a user installation established some other default when the PL/I Transient Library was installed. *Setting the proper ISASIZE for every projection MVS PL/I program and every DOS/VS PL/I CICS/VS transaction is the most important single thing one can do to optimize PL/I program performance!*

15.2.3.2 REPORT

This option allows you to obtain statistics on the memory usage by PL/I during execution of the program. Refer to 15.11, “Storage Management in PL/I” on page 345. Use the output of the REPORT option to set ISASIZE correctly. Do ***not*** run programs in production with this option turned on. It is expensive.

15.2.3.3 COUNT FLOW

It is possible at execution to suppress the COUNT and FLOW options requested at compilation. This is useful as it is possible to have a program in production, compiled with these options, but turn them off at execution time. If a problem arises, it will then be possible to repeat the execution of the program without having to recompile it. Normally, however, these options are ″compiled out″ of production programs, since they add overhead in the executable code.

15.2.3.4 SPIE STAE

As user-program execution options they authorize PL/I to issue SPIE and STAE macros to intercept program checks and abends. It is possible with NOSPIE and NOSTAE to prevent this and in this case it is no longer certain that the management of errors will be handled by system ABEND or by an interruption handling program. This, therefore, allows an error routine to call PL/I modules and to continue to secure to itself the management of errors.

15.2.4 The EXEC and PROCESS Cards

It is possible to pass information to the PL/I compiler either by the EXEC or PROCESS statement. The PROCESS statement has the advantage of being processed by the PL/I compiler and therefore does not follow the OS JCL conventions. It can, as for all PL/I statements, be contained in more than one source program line and is not therefore limited in length. The parameter of the EXEC statement, on the other hand, is limited to a maximum of 100 characters. The format of the PROCESS statement is the same in MVS and DOS.

15.3 Linkages Between Languages

15.3.1 Linkages Supported

The linkages supported in MVS PL/I are those with COBOL, FORTRAN and Assembler. The support is obtained by specifying the necessary information in the PROC and ENTRY statements, refer to the respective Programmers′ Guides. In the case of Assembler, it is necessary to guard against modifying the contents of register 12 if PL/I is to intercept ABENDs or program checks. Modification of register 12 can cause relatively distracting messages such as:

```
ERROR IN PL/I ERROR HANDLER
```

This will occur if a program check or ABEND occurs in the Assembler subroutine at a time when R12 does not point to the PL/I control block expected by the PL/I error handler.

15.3.2 Linkages not Supported

The linkage with RPG II is supported in DOS PL/I. The MVS PL/I compiler does not support linkage with RPG II.

15.4 ENVIRONMENT Attributes

Most of the ENVIRONMENT options are no longer necessary in MVS, as they are required only at execution time and are therefore specified in the DCB parameter of the DD statement.

15.4.1 Not Supported in MVS

15.4.1.1 MEDIUM

Physical and logical unit type. This option is ignored by the MVS compiler. The error severity is 8 and gives correct execution. In MVS PL/I the name of the DD statement (DDname) is the name of the file as specified in the DCL. If some other name must be used, it can be supplied via the TITLE option on the OPEN statement.

15.4.1.2 FUNCTION

This option defines the type of operation to be carried out on a 3525, 2560 or 5425. It is ignored in MVS. The error severity is 8 and gives correct execution. If the function exists in MVS, it will be specified in the sub-parameter FUNC of the DCB parameter of the DD statement.

15.4.1.3 ASSOCIATE

This option corresponds to the multiple functions on the 3525. The functions required will be specified in the JCL by the sub-parameters FUNC and AFF of the DCB parameter in the DD statement.

Note: The use of this parameter in MVS PL/I causes an error of severity 12 (serious). Use of this option prohibits the use of spooling.

15.4.1.4 RCE (Read Column Eliminate)

This option is specified in the DCB by the parameter MODE=ER or MODE=CR. A statement FORMAT (nl,n2),(n3,n4) must precede the data in the input stream. Use of this option prohibits the use of spooling.

15.4.1.5 OMR (Optical Mark Read)

This option disappears and re-appears in the DCB: MODE= EO or MODE=CO. It is used for reading optical marks on the 3525, and prohibits the use of spooling.

15.4.1.6 COLBIN

This option is suppressed in MVS PL/I and the equivalent function is supplied to MVS by the sub-parameter MODE=C in the DCB.

15.4.1.7 STACKER

This option is withdrawn from the MVS compiler and supplied by using ASA or machine control characters: CTLASA or CTL360 in PL/I, or equally well RECFM=FBA or RECFM=FBM in the DCB.

15.4.1.8 CMDCHN WTRPROT FILESEC NOFEED VOLSEQ

Options for handling 3540 diskettes, which are not supported by MVS PL/I.

15.4.1.9 UNLOAD

This option does not exist in MVS. The unloading of tapes after a CLOSE is determined by the DISP= parameter of the DD statement.

15.4.1.10 NOTAPEMK NOLABEL

These are specified in the JCL in the LABEL parameter of the DD statement.

15.4.2 Supported but to be Avoided

In OS most of the environment parameters can be specified in the DCB. At OPEN, MVS merges the information from the program, from the label if the file exists already, and from the DCB parameters in the DD statement. It is therefore damaging to specify the physical blocksize in the program, because a re-compilation is involved if the blocking factor is to be changed. The following options should therefore be omitted:

```
BLKSIZE
KEYLENGTH
KEYLOC
```

15.4.3 The ″TOTAL″ Option

A new option can be specified: TOTAL. This option, which is effective only with CONSECUTIVE files, allows PL/I to branch directly to MVS access method routines without using the TRANSIENT library modules; there is therefore a performance improvement. This requires that the file be declared as completely as possible. Only the blocking factor may be specified in the DD statement, and no options may be specified at OPEN. Users must weigh the benefits of improved performance (via the TOTAL option) against the advantages of complete MVS DCB merge.

15.4.4 The SIS Option (Sequential Insert Strategy)

This option applies to (and only to) processing of a VSAM KSDS using a PL/I file with the DIRECT attribute. It causes VSAM to insert new records using SIS (Sequential Insert Strategy) rather than direct insert strategy. All other environment options, in the case of VSAM files, are identical in MVS PL/I and DOS PL/I.

15.5 Calling SORT from PL/I

15.5.1 Interfaces Offered

The DOS PL/I Optimizer provides, through PLISRTx, an interface to DOS/VS SORT/MERGE Version 2 (5746-SM2) and other VSE supported sorts.

The MVS PL/I Optimizer offers, through an interface of the same name, access to DFSORT (5740-SM1).

The four sort entry points offered by PL/I are the same in MVS and VSE: PLISRTA, PLISRTB, PLISRTC and PLISRTD. The only exits supported by PL/I are E15 and E35.

15.5.2 Parameters to be Passed

The parameters for calling sort are fortunately the same. The number of these parameters depends on the entry point used. Let us take the most general case (PLISRTD) which invokes exits E15 and E35. The parameters to be passed are described in the following sections.

15.5.2.1 SORT FIELDS

SORT data: a character string containing an image of the SORT statement. This card image must begin and end with a blank. It will contain the sort criteria and the description of these criteria.

15.5.2.2 RECORD

RECORD information: A character string containing a card image of the RECORD control statement. It too must begin and end with a blank. It describes the length and format of the records.

15.5.2.3 STORAGE

STORAGE information: This will be a FIXED DECIMAL expression indicating the amount of storage available to the sort.

15.5.2.4 RETURN CODE

This will be a FIXED BINARY variable where the sort will place a return code equal to 0 or 16 according to the correct or incorrect result of its execution.

15.5.2.5 E15 EXIT PROCEDURE

The name of the PL/I function procedure to be executed at the sort input exit point is E15. It will pass to the sort one-by-one the records to be sorted. Note that when the sort is called from PL/I, it does not allow merging of records read from SYSIN with records passed from the E15 exit procedure.

15.5.2.6 EXIT E35

Name of the entry point is F35, and name of the PL/I procedure (internal or external) which will receive control after the sort phase.

15.5.2.7 DDNAME PREFIXES

The names of the sort files are defined by default by the sort. The first four characters can be defined by the user:

```
SORTIN XXXXIN
SORTOUT XXXXOUT
SORTWKO1 XXXXWKOI
SORTWKOn XXXXWKOn
SORTCKPT XXXXCKPT
```

The other files must remain as SYSLIB and SYSOUT. This parameter therefore allows the user to define DDnames convenient to him.

15.5.2.8 SORT MESSAGES

The user can ask for sort messages to be directed to the console operator or to the SYSOUT file, and to specify the severity of the messages:

```
NO   No messages on SYSOUT
AP   All messages on SYSOUT
AC   All messages to the console
CP   Critical messages on SYSOUT
CC   Critical messages on the console
```

15.5.2.9 SORT TECHNIQUES

The user can specify a particular sort technique:

```
PEER   Peerage sort
BALN   Balanced
CRCX   Criss-cross
OSCL   Oscillating
POLY   Polyphase
```

The sort call is, therefore, little different to that of DOS PL/I. In all cases, reading the Programmers′ Guide for the appropriate version of the sort is recommended.

15.6 Checkpoint-Restart in PL/I

15.6.1 PLICKPT

Whereas the DOS checkpoint authorizes only manual restart with operator intervention, in MVS it is possible to request it by means of JCL and if the program lends itself to automatic restart, it will restart at the last step or the last checkpoint.

There are always four parameters to be supplied:

```
CALL PLICKPT (p1,p2,p3,p4); (DOS and MVS)
```

but the third has functional differences. In DOS the parameter defines the physical and logical unit on which the checkpoints will be written (SYS00x..,33xx). In MVS this defines the checkpoint organization (sequential or partitioned). For the other parameters, there is little or no difference:

```
P1: DLBL in DOS, DDNAME in MVS.
P2: Name to be given to the checkpoint by the operating system. This
    name will be specified in control statement on a deferred restart.
P4: Return code returned by the checkpoint routines after execution.
    The values of these return codes are compatible between the two
    compilers.
```

15.6.2 PLIREST

An additional function offered by MVS is available in PL/I; automatic restart. This is possible thanks to a new function of the MVS PL/I optimizer: PLIREST. This includes use of ABEND U4092. As long as this code is in the list of eligible ABEND codes specified at system generation, an automatic restart will be possible. If there is to be automatic restart, the operating system must be able to recognize an abnormal end. With PL/I intercepting program checks, the end of execution appears to the system as ′normal′. Thus the function of PLIREST is to induce an ABEND. It is, therefore. necessary to code some instructions of the type:

```
IF ONCODE= xxx THEN DO;
  ...........:
  ...........:
CALL PLIREST;
END;
```

On restart, control will be returned to PL/I after the instruction CALL PLICKPT (.....). On testing the return code, the programmer will be able to tell if he has a restart or after writing a checkpoint.

15.6.3 PLICANC

Another possibility offered in the MVS PL/I optimizer is the ability to annul a request for an automatic restart. The function PLICANC provides this service.

In checkpoint-restart MVS PL/I offers more facilities than DOS PL/I. The conversion of PL/I programs poses few syntax problems. On the other hand, to take advantage of automatic restart, some additional work will be necessary in the logic of the program.

15.7 DUMP in PL/I Optimizer

15.7.1 Output File

The PL/I optimizer possesses its own dump routines. They present a number of advantages over the dumps provided by the operating system. Among others, they trace the PL/I control blocks (TCA, DSA and so on). In MVS PL/I optimizer, the dump is edited on a particular file whose DDNAME is PLIDUMP. In DOS this dump is produced on SYSLST. It is therefore necessary to provide a //PLIDUMP DD control statement in the JCL. If this is absent, the dump is not produced and a message is produced by MVS indicating that

```
"FILE PLIDUMP COULD NOT BE OPENED - DDNAME MISSING".
```

Note that the PLIDUMP file is also used for data requested by the COUNT and REPORT options. Do not suppress the storage report produced by the REPORT option by omitting the PLIDUMP DD card; use NOREPORT to suppress this report.

15.7.2 Options Specific to DOS

The option D (and ND), which asks for information on the opened files and on the modules associated with them, does not exist in MVS. Only the option F (and NF) exists. In practice this trace of called modules is necessary in DOS, because the loading of modules from the TRANSIENT LIBRARY is done by PL/I, which implies that only PL/I knows which modules are loaded. In MVS, the load list keeps track of loaded modules and the option PLIDUMP (TRACE) produces a list of these modules.

The options 48 and 60 request the PL/I dump routines to use different translation tables; there is no equivalent in MVS.

The option R (and NR), which in DOS allows the collection of data on the management of storage, is not required in MVS while calling DUMP, but is replaced by the execution option REPORT (and NOREPORT).

The option Q does not exist and has no equivalent in MVS. In DOS it allows a more succinct dump, consisting of a DOS PDUMP of all of PL/I's storage. It is used when the 10K or so of storage needed by normal PL/I DUMP is not available.

15.7.3 Options Specific to MVS

The options A, E and 0 are used only in a multitasking environment:

```
A  Dump all tasks
O  Dump only the task requesting the dump
E  Use of an exit
```

15.7.4 Compatibility

Parameters unsupported by the PL/I dump routines are ignored if they are used when calling dump facilities.

15.8 Return Codes in PL/I

15.8.1 Setting Return Codes

It is possible to set return codes in PL/I. This capability, which already existed in DOS when returning from a sub-program or a function, is now extended to main programs. The function PLIRETC allows the setting of a return code. For example, CALL PLIRETC (16); will set the return code to 16, thus allowing JCL to test it by COND parameters in succeeding STEP statements.

15.8.2 Return Code Values

Note nevertheless the fact that the code returned from a PL/I program is the SUM of the return codes provided by the user in the call to the function PLIRETC (0 by default) and a code determined by the PL/I termination routines indicating how the job terminated:

```
0000 Normal end
1000 STOP, EXIT, PLIDUMP('S'), PLIDUMP('E'), insufficient storage.
2000 ERROR condition and no ON ERROR or ON FINISH block.
4000 ERROR in PL/I management routines.
```

15.9 Forcing an ABEND

It can be useful to force PL/I to end in ABEND in certain cases. In practice, while a user's program may end abnormally, a PL/I program nevertheless ends quite normally as far as the operating system is concerned. This can be accomplished by executing the program with the NOSTAE option, writing an IBMBEERA routine (see Programmer's Guide and Execution Logic), or writing an assembler routine to cancel PL/I's STAE macro and issue the ABEND macro.

15.9.1 Use of DISP in the JCL

It is possible in the DISP parameter of the JCL statement to specify for example: DISP=(NEW,CATLG,DELETE). The third parameter of DISP is used to indicate to the operating system which route to follow in case of an ABEND. If PL/I intercepts and handles errors, the system does not see an ABEND condition and the file will be cataloged (second parameter) and not destroyed. It is therefore necessary to force an ABEND. There are two possibilities:

- Reassemble the module IBMBEERA to load register 15 with a non-zero value.
- Use the PLIREST function to cause a user ABEND.

15.9.2 Automatic Restart

An automatic restart in the case of an ABEND can only take place if an ABEND is actually detected; it must therefore be forced. This is the role of PLIREST.

15.10 Overlay Structures

Overlay structures defined in DOS PL/I optimizer programs can remain valid in MVS PL/I optimizer programs. However, the CALL PLIOVLY statements must be removed. MVS linkage editor facilities (PARM=OVLY, INSERT, and OVERLAY) are used to build the overlay structure as discussed below.

15.10.1 Conversion

When converting a DOS PL/I program to MVS, if the former used an overlay structure, it will be necessary to suppress all calls to PLIOVLY, which do not make sense in MVS. The overlay structure will therefore disappear, and in the majority of cases will no longer be necessary in MVS, each user having his own independent address space. Programs which rely on REFETCH of overlays to re-initialize static storage will have to remain as overlay programs or be modified.

15.10.2 Overlay in MVS

If it is still necessary to retain an overlay structure, it will be necessary to link-edit the program specifying PARM.LKED= OVLY and providing the linkage editor with INSERT and OVERLAY control statements. This is completely independent of the PL/I source code but not of the logic of the program.

15.11 Storage Management in PL/I

15.11.1 Storage Management in DOS

In DOS, PL/I considers that all the storage available in the partition is dedicated to it. It separates storage requests into LIFO and non-LIFO. DSA allocations are LIFO requests, in particular for AUTOMATIC variables. These requests are considered as LIFO because storage is de-allocated in the reverse order to allocation. Non-LIFO requests include, among others, the allocation of CONTROLLED and BASED variables, as are requests for space for loading transient modules, and for SORT if PLISRTx is used.

15.11.2 Storage Management in MVS

In MVS, PL/I acquires an Initial Storage Area (ISA). Its size is set by default when the user installs the PL/I Transient Library. The IBM-supplied default is half the storage outside the load module. The rest is left for the operating system. The distinction between LIFO and non-LIFO remains, but note that loading of modules from the transient library is done by MVS, and this will use the storage left free by PL/I. If the storage allocated to PL/I is not enough at the start of execution, it will issue GETMAINs for that part left for the operating system. These GETMAINs can slow down the execution of the program. The amount of storage allocated by PL/I at the start of execution and the number of GETMAINs and FREEMAINs can be found by use of the REPORT option. The amount of storage space allocated by PL/I at the start of execution can be specified by the parameter ISASIZE. It is almost always worthwhile to give a PL/I program a

large enough ISA to reduce subsequent GETMAINs and FREEMAINs to zero (or some very small number). This is one of the most important things a user can do to improve the performance of a PL/I program.

15.12 PL/I and CICS

15.12.1 File Support

The only PL/I file supported by PL/I under CICS is SYSPRINT, and its usage must be limited. The functions LINENO and COUNT are supported under MVS, but give a null value if they are used under DOS. Other file access facilities are provided by CICS/VS file access macros and/or EXEC commands.

15.12.2 Statements not Supported

DISPLAY and DELAY are not supported. The functions PLIREST, PLICANC, PLICKPT and PLISRTx are not supported. MULTITASKING is not supported. Inter-language linkages are limited to Assembler (with some restrictions).

15.12.3 CALLing DUMP

The only options permitted in calling DUMP are T and NT, S and C, B and NB, K and NK. The dump is not written to the PLIDUMP file, but is sent by transient files to a destination of CPLD. K and NK are special CICS/VS-only options on PLIDUMP.

15.12.4 Execution Options

The execution options can only be communicated to PL/I by means of a PLIXOPT string:

```
DCL PLIXOPT CHAR(nn) VAR EXT INIT('xuxuxx');
```

The options permitted are COUNT and NOCOUNT, FLOW and NOFLOW, ISASIZE, REPORT and NOREPORT, STAE and NOSTAE. The option SPIE will be ignored if it is specified. If a PLIXOPT string is not specified, defaults will be supplied. If they are wrong, the mistake can be very costly in CICS/VS transaction execution time. Every PL/I-CICS/VS transaction program should have a PLIXOPT string, especially to set ISASIZE. Note that the STAE option causes PL/I to issue a CICS DFHPC SETEXIT macro. It uses CICS error handling. It does not override it.

15.12.5 Compatibility

All these restrictions already exist in DOS. No differences exist at the PL/I level. The compatibility of CICS transactions written in PL/I appears therefore to be total.

15.12.6 PL/I-CICS/VS Transaction ABEND Codes

These are documented in the PL/I Programmer's Guide, not the CICS manuals. The only one which tends to puzzle users is "APLS", which means that the PL/I ERROR condition was raised, but *not by a program check or a transaction abend.* For example, CONVERSION might have been raised by the PL/I library.

15.12.7 PL/I Return from ON-units and CICS Transaction Backout

If a CICS transaction ABEND begins but is intercepted in a PL/I ON-unit, and the program takes ″normal return″ from the ERROR ON-unit (that is, quits) PL/I will reissue the CICS ABEND. If ERROR was raised *without* a CICS ABEND and the program takes ″normal return″ from the ERROR ON-unit, PL/I issues the APLS ABEND. (See previous paragraph.) In either case, the ABEND can drive CICS Dynamic Backout, or it can raise ERROR (with ONCODE 8090, ″An ABEND has occurred.″) in an earlier PL/I program that invoked the failing one via a CICS LINK.

Chapter 16. FORTRAN

16.1 VS FORTRAN in OS/390

VS FORTRAN is the compiler and library to use on OS/390. VS FORTRAN expands greatly on what you can do with FORTRAN in accessing system services and/or hardware features. If you have used VS FORTRAN on VSE, you may be aware of the extensions that VS FORTRAN provides over DOS FORTRAN, such as execution time dynamic commons, compile-time included source files, asynchronous I/O, and level 66 language compatibility. VS FORTRAN has multitasking on OS/390, and the Version 2 product offers programming enhancements such as structured programming constructs, long variable names, an Intercompilation Analyzer, and an interactive full screen debugger. In Version 2 Release 3, data-in-virtual support and dynamic file allocation are available. The Version 2 product also supports the IBM 3090 vector hardware, and it also conforms to the SAA Common Programming Interface (CPI).

16.2 FORTRAN Conversion Considerations

The conversion to OS/390 means that some changes are needed to certain DOS FORTRAN programs. First, the OPSYS routine is no longer supported. The I/O services and more are supplied in Version 2 Release 3 with dynamic file support. The overlay support provided by OPSYS is no longer needed, because now larger programs are supported in the 2 Gigabyte address spaces supported by OS/390. Options that were used before to control compilation are now handled in part by cataloged procedures and in part by the compiler options VS FORTRAN now recognizes. VS FORTRAN also now allows execution time options to control the running of your production programs.

Although recoding of programs is not necessary except for routines using OPSYS, the benefits to your programs can be measured in increased performance due to the many optimization levels the compiler provides. Exploitation is available to recompiled programs so that they may now run above the 16MB line. Access to the vector hardware is available through recompilation as is access to new operating system features.

Chapter 17. Language Environment (LE)

17.1 Introduction

This chapter introduces OS/390 Language Environment (program number 5645-001). OS/390 Language Environment is the language run-time environment distributed with OS/390.

Various strategies for migrating your applications to the Language Environment run-time are considered. These strategies depend on the programming language, the version of VSE you use, and whether you already use LE/VSE.

The information presented in this chapter is not sufficient by itself to carry out a successful migration to OS/390 Language Environment. You should study carefully the publications referred to in Table 35 on page 353 for more information. This chapter is intended to draw attention to the more obvious problems that can arise in such a migration.

17.1.1 General Comments on Language Environment

OS/390 Language Environment is the run-time environment you receive when you order your OS/390 system software.

OS/390 Language Environment provides common services and language-specific routines in a single run-time environment for C, C++, COBOL, FORTRAN, PL/I, and Assembler applications. It offers consistent and predictable results for language applications, independent of the language in which they are written.

If you are migrating to OS/390 Language Environment from a non-Language Environment run-time environment, you should read the *OS/390 Language Environment Concepts Guide* to understand the concept of Language Environment.

17.1.1.1 A Few Words about COBOL and PL/I

With the many different environments and language products (COBOL and PL/I) here is a table to help you understand where you can run your COBOL and PL/I products.

Table 34. COBOL and PL/I: What Runs Where?

Host Operating System	Host COBOL and PL/I Products	Run-Time Library Support
MVS 4.3 through MVS 5.2.2	COBOL for MVS & VM PL/I for MVS & VM	Language Environment for MVS & VM Rel 5
OS/390 Ver 1 Rel 1,2	COBOL for MVS & VM PL/I for MVS & VM	Language Environment element of OS/390
OS/390 Ver 1 Rel 3 OS/390 Ver 2 Rel 4,5	COBOL for OS/390 & VM COBOL for MVS & VM PL/I for MVS & VM	Language Environment element of OS/390
VSE/ESA Ver 1 Rel 4 VSE/ESA Ver 2 Rel 1,2,3	COBOL for VSE/ESA PL/I FOR VSE/ESA	Language Environment for VSE/ESA Rel 4

17.1.2 Conceptual Differences between LE/VSE and OS/390 Language Environment

There are some conceptual differences between LE/VSE and OS/390 Language Environment. These differences do not affect the running of your migrated LE/VSE applications in an OS/390 Language Environment but you may want to take advantage of the extra facilities offered by the OS/390 Language Environment. For more information, refer to the *LE/VSE Concepts Guide Release 4*, or *LE/VSE Concepts Guide Release 1*, and the *OS/390 Language Environment Concepts Guide*.

- OS/390 Language Environment supports multithreading. LE/VSE supports only single threading.
- OS/390 Language Environment supports applications consisting of one or more processes. LE/VSE supports only a single process for each application that runs in the common run-time environment.
- OS/390 Language Environment supports multiple threads within an enclave. LE/VSE supports only a single thread within an enclave.

17.2 VSE to OS/390 Migration Considerations

The strategy you follow to migrate your run-time environment to OS/390 depends on the programming language, the run-time environment you are using in VSE, and the version of VSE you are running.

If you are using an LE/VSE-conforming language, you may be able to transfer your compiled object code to OS/390 from your VSE system, link-edit it with OS/390 Language Environment and run it there without further change. See below for a list of LE/VSE-conforming languages.

Whatever language, run-time environment or version of VSE you are running, you will at least have to relink your object code in OS/390. It is not possible to transfer phases from your VSE libraries to OS/390.

17.2.1 LE/VSE-conforming Languages

An LE/VSE-conforming language is any high-level language (HLL) that adheres to the LE/VSE common interface.

There are three LE/VSE-conforming languages:

C for VSE/ESA	program number 5686-A01
COBOL for VSE/ESA	program number 5686-068
PL/I for VSE/ESA	program number 5686-069

These languages require LE/VSE to be available at compile-time, as well as link-edit and run-time. LE/VSE requires VSE/ESA version 1 release 4, or VSE/ESA version 2 or later. C for VSE/ESA requires LE/VSE 1.4. You cannot compile or run C for VSE/ESA programs under LE/VSE 1.1.

Any HLL not listed above, is known as a non-LE/VSE-conforming language. These include C/370, DOS/VS COBOL, VS COBOL II, DOS PL/I and VS FORTRAN.

17.2.2 Useful Publications

Table 35 lists some publications that you may find useful when planning your conversion.

Table 35. Useful Publications

Publication Title	Form Number
OS/390 Language Environment Migration Guide	SC28-1944
OS/390 Language Environment Programming Reference	SC28-1940
OS/390 Language Environment Programming Guide	SC28-1939
OS/390 Language Environment Concepts Guide	GC28-1945
OS/390 Language Environment Customization	SC28-1941
OS/390 C/C++ V2R4.0 Programming Guide	SC09-2362
Language Environment V1R5 FORTRAN Migration Guide	SC26-8499
LE/VSE Programming Guide Release 4	SC33-6684
LE/VSE Programming Guide Release 1	SC26-8065
LE/VSE Programming Reference Release 4	SC33-6685
LE/VSE Run-Time Migration Guide Release 4	SC33-6687
LE/VSE Concepts Guide Release 4	GC33-6680
LE/VSE Concepts Guide Release 1	GC26-8063
VSE/ESA Enhancements	SC33-6629
Taking Advantage of IBM Language Environment for VSE/ESA	SG24-4798
COBOL for OS/390 & VM Complier and Run-Time Migration Guide	GC26-4764
IBM PL/I for MVS & VM Compiler and Run-Time Migration Guide Release 1.1	SC26-3118
OS/390 C++ Compiler and Run-Time Migration Guide	SC09-2359

17.3 Migrating from LE/VSE-Conforming Languages

This section discusses briefly how you can migrate LE/VSE-conforming language applications to OS/390 Language Environment. You should also read 17.5, “Migrating from LE/VSE” on page 359 for more information.

17.3.1 C for VSE/ESA

Even though C for VSE/ESA is an LE/VSE-conforming language, you cannot transfer your C/VSE compiled object code to OS/390, link-edit it and expect it to run. You must recompile it with OS/390 C/C++. However, C/VSE source code is generally compatible with OS/390 C/C++, so your C/VSE programs should compile under OS/390 C/C++ with minimal changes. Refer to 17.4.2, “C/370” on page 355 for information on migrating your C/VSE applications to OS/390.

17.3.2 COBOL for VSE/ESA

COBOL for VSE/ESA is an LE/VSE-conforming language. If your COBOL applications are written in COBOL/VSE, they can (subject to certain restrictions) be migrated to OS/390 without change. You can transfer the compiled object code from VSE to OS/390, link-edit it with OS/390 Language Environment and run it there. This is discussed in 12.2, "VSE to OS/390 Migration Considerations" on page 250.

17.3.3 PL/I for VSE/ESA

Even though PL/I for VSE/ESA is an LE/VSE-conforming language, you cannot transfer your PL/I VSE compiled object code to OS/390, link-edit it and expect it to run. You must recompile it with PL/I for MVS and VM. However, PL/I VSE source code is compatible with PL/I for MVS and VM, so your PL/I VSE programs should compile under PL/I for MVS and VM without change. Refer to Chapter 15, "PL/I" on page 333 for information on migrating your PL/I VSE applications to OS/390.

17.4 Migrating from Non-LE/VSE Run-time Environments

This section discusses some of the considerations of which you should be aware, if you are migrating to OS/390, and therefore OS/390 Language Environment, from a non-LE/VSE run-time environment.

If you are running VSE/ESA version 1 release 4 or VSE/ESA version 2, but you are not using LE/VSE, you should consider implementing LE/VSE in your VSE system, before migrating your run-time to OS/390. This may require that you also implement a new version of your language compiler. However, it may be easier to convert to a new version of compiler and run-time in the VSE environment, which is familiar to you, than to convert to a new compiler, run-time and operating system, all at the same time. Refer to the relevant chapters in this book on migrating COBOL, C and PL/I applications to OS/390.

17.4.1 Options Mapping

Details of the mapping of options in OS/390 Language Environment, are to be found in the *OS/390 Language Environment Migration Guide*. Mapping of options for LE/VSE 1.4 are described in the *LE/VSE Programming Reference Release 4*. You can also find tables comparing the use of options in DOS PL/I, C/370, DOS/VS COBOL, or VS COBOL II, and in LE/VSE 1.4, in the *LE/VSE Run-Time Migration Guide Release 4*.

In general the mapping of options from these non-LE/VSE-conforming languages, as described in these tables, is the same for OS/390 Language Environment as for LE/VSE. However, if you are migrating from DOS PL/I or C/370 with LE/VSE you should also consider the information about the REPORT and ISASIZE options in Table 36 on page 355.

Table 36. REPORT and ISASIZE Options, C/370 and DOS PL/I	
REPORT option, C/370 and DOS PL/I	The information supplied by the REPORT option in C/370 and DOS PL/I is supplied in LE/VSE by the RPTSTG option. The RPTOPTS option may also be of use in determining storage use.
ISASIZE option, DOS PL/I	In LE/VSE 1.4 the ISASIZE option maps to the STACK option. In OS/390 Language Environment ISASIZE maps to STACK, NONIPTSTACK and PLITASKCOUNT.

17.4.2 C/370

C/370 is not an LE/VSE-conforming language. If your applications are written in C/370, you must convert them to another version of C before you can run them under OS/390. A well-coded C/370 application should generally recompile successfully with OS/390 C/C++, and run successfully with OS/390 Language Environment without modification.

Table 37 lists some migration considerations you should be aware of when migrating from C/370.

Table 37. C/370 Migration Considerations	
Migration Consideration	**Comments**
Standard Streams	In C/370, you could override the destination of error messages by redirecting `stderr`. OS/390 Language Environment determines the destination of all messages from the MSGFILE run-time option.
Passing Command Line Parameters	In C/370 if an error was detected with the parameters being passed to the main program, the program terminated with a return code of 8 and a message indicating the reason the program terminated. Under OS/390 Language Environment the same message is displayed, but the program also terminates with a 4093 abend, reason code 52 (hexadecimal 34).
Prefix of perror() and strerror() Messages in C	With OS/390 Language Environment all `perror()` and `strerror()` messages in C contain a prefix. With C/370, there was no prefix on these messages. The prefix is EDC*xxxxa* where *xxxx* is a number (always *5xxx*) and *a* is either I, W, or E.
Storage Report	The format of the run-time storage report generated by the OS/390 Language Environment RPTSTG run-time option is different from the format of the storage reports produced by the C/370 REPORT run-time option.

17.4.3 VS COBOL II

VS COBOL II is not an LE/VSE-conforming language. However, VS COBOL II applications may run with OS/390 Language Environment with minimal changes. Subject to certain restrictions, you can transfer your VS COBOL II compiled object code from VSE to OS/390, link-edit with OS/390 Language Environment and run it there. This is discussed in Chapter 12, “COBOL” on page 249.

Table 38 on page 356 lists some migration considerations you should be aware of when migrating from VS COBOL II.

Table 38. VS COBOL II Migration Considerations

Migration Consideration	Comments
Abends	In VS COBOL II, a severe unhandled error condition always resulted in an abend. With OS/390 Language Environment, you use the ABTERMENC run-time option to specify whether a severe unhandled condition results in an abend or a normal termination with a return code and reason code. The IBM-supplied installation value for the ABTERMENC run-time option is ABTERMENC(RETCODE). To ensure that your application ends with an abend when there is a severe unhandled condition, specify ABTERMENC(ABEND).
Storage Report	The format of the run-time storage report generated by the OS/390 Language Environment RPTSTG run-time option is different from the format of the storage reports produced by the VS COBOL II SPOUT run-time option.

17.4.4 DOS/VS COBOL

DOS/VS COBOL is not an LE/VSE-conforming language. If your applications are written in DOS/VS COBOL, you must update the source and compile with COBOL for MVS & VM or COBOL for OS/390 & VM before you can run them under OS/390. Refer to Chapter 12, “COBOL” on page 249 for further information.

Table 39 lists some migration considerations you should be aware of when migrating from DOS/VS COBOL.

Table 39. DOS/VS COBOL Migration Considerations

Migration Consideration	Comments
Abends	In DOS/VS COBOL, a severe unhandled error condition always resulted in an abend. With OS/390 Language Environment, you use the ABTERMENC run-time option to specify whether a severe unhandled condition results in an abend or a normal termination with a return code and reason code. The IBM-supplied installation value for the ABTERMENC run-time option is ABTERMENC(RETCODE). To ensure that your application ends with an abend when there is a severe unhandled condition, specify ABTERMENC(ABEND).

17.4.5 DOS PL/I

DOS PL/I is not an LE/VSE-conforming language. If your applications are written in DOS PL/I, you must update the source and compile with PL/I for MVS & VM before you can run them under OS/390. Refer to Chapter 15, “PL/I” on page 333 for further information.

Table 40 on page 357 lists some migration considerations you should be aware of when migrating from DOS PL/I.

Table 40. DOS PL/I Migration Considerations

Migration Consideration	**Comments**
Dumps	The output produced by PLIDUMP is different when running under OS/390 Language Environment.
Condition Handling	In general, PL/I condition handling continues to function in the same way when running under OS/390 Language Environment; however, you should consider the following: • The ERRCOUNT run-time option specifies how many conditions of severity 2, 3, and 4 can occur before the enclave terminates abnormally. The IBM-supplied installation value for the ERRCOUNT run-time option is ERRCOUNT(20). This value is not suitable for all PL/I applications. To ensure that your application behaves correctly, and is compatible with DOS PL/I behavior, specify ERRCOUNT(0). • The diagnostic message for an ERROR condition is issued only if there is no ERROR ON-unit established, or if the ERROR ON-unit does not recover from the condition by using a GOTO out of block. However, for other PL/I conditions whose implicit action includes printing a message and raising the ERROR condition, the message is issued before control is given to an established ERROR ON-unit.
Run-Time Message Output - SYSPRINT	OS/390 Language Environment directs run-time message output from PL/I programs to the file specified by the OS/390 Language Environment MSGFILE run-time option, instead of to the PL/I SYSPRINT file. User-specified output is still directed to the PL/I SYSPRINT file. If you want OS/390 Language Environment to handle this output, specify the run-time option MSGFILE(SYSPRINT). When you specify MSGFILE(SYSPRINT), SYSPRINT contains both run-time messages and user-specified output.
Format and Content of Messages in PL/I	The format and content of run-time messages is different for PL/I applications running with OS/390 Language Environment. If you have applications that analyze run-time output, you should change them. Differences include: • The message number in the message prefix is now four digits instead of three digits. • The message severity in the message prefix can now be I, W, E, S, or C. • The message text of some mixed-case English and Japanese messages has been enhanced.
DEPTHCONDLMT Option	The default setting for the DEPTHCONDLMT option, both for CICS and non-CICS, is DEPTHCONDLMT(10). The recommended setting for PL/I applications, for compatibility with DOS PL/I, is DEPTHCONDLMT(0).
Storage Report	The format of the run-time storage report generated by the OS/390 Language Environment RPTSTG run-time option is different than the format of the storage reports produced by the DOS PL/I REPORT run-time option.

17.4.6 VS FORTRAN

If your VSE applications are currently written in VS FORTRAN, you must convert them to another version of the FORTRAN compiler before you can run them under OS/390. There is currently no LE/VSE-conforming FORTRAN compiler, so you must convert your VS FORTRAN applications to the OS/390 version of VS FORTRAN. You should read the *Language Environment V1R5 FORTRAN Migration Guide* for information about migrating to Language Environment-enabled FORTRAN.

17.4.7 Migrating Interlanguage Communications Applications

Interlanguage communications (ILC) applications are applications built of two or more high-level languages (HLLs), and, frequently, Assembler. ILC applications run outside the realm of a single language′s environment, which creates special conditions, such as how each language maps data, how conditions are handled, or how data can be called and received by each language.

If your ILC applications are built only of two or more LE/VSE-conforming HLLs, then migrating them to OS/390 Language Environment is the same as migrating applications in one LE/VSE-conforming language. This section considers the migration of ILC applications with two or more non-LE/VSE-conforming language.

Table 41 gives information about the migration of ILC applications with various combinations of non-LE/VSE-conforming languages.

Table 41 (Page 1 of 2). ILC Migration Considerations

To Migrate:	**You Need To:**
A phase containing one or more DOS/VS COBOL programs, with calls to or from DOS PL/I	1. Upgrade the DOS/VS COBOL source code, and compile with COBOL for OS/390 and VM or COBOL for MVS & VM. 2. Upgrade the DOS PL/I source code, and compile with PL/I for MVS and VM. 3. Link-edit the load module with OS/390 Language Environment.
A phase containing one or more VS COBOL II programs, with calls to or from DOS PL/I	1. Upgrade the DOS PL/I source code, and compile with PL/I for MVS and VM. 2. Transfer the VS COBOL II object code to OS/390. 3. Link-edit the load module with OS/390 Language Environment.
A phase containing one or more DOS/VS COBOL programs, with calls to or from C/370	1. Upgrade the DOS/VS COBOL source code, and compile with COBOL for OS/390 and VM or COBOL for MVS & VM 2. Upgrade the C/370 source code, and compile with OS/390 C/C++. 3. Link-edit the load module with OS/390 Language Environment.

Table 41 (Page 2 of 2). ILC Migration Considerations	
To Migrate:	**You Need To:**
A phase containing one or more VS COBOL II programs, with calls to or from C/370	1. Upgrade the C/370 source code, and compile with OS/390 C/C++. 2. Transfer the VS COBOL II object code to OS/390. 3. Link-edit the load module with OS/390 Language Environment.

17.4.8 Migrating Assembler Applications

The Assembler distributed with OS/390 is the High Level Assembler for MVS & VM & VSE (HLASM). Therefore, this is the same product as the high level Assembler distributed with VSE/ESA. If you are using the old VSE Assembler, you will first have to convert your assembler programs to HLASM. You can use HLASM on VSE/ESA version 1 release 2 or later.

To use HLASM with LE/VSE or OS/390 Language Environment, you need to code the applications with the assembler macros provided with LE/VSE or OS/390 Language Environment. You must also ensure that the assembler programs adhere to certain conventions for register and storage usage, for condition handling and accessing input parameters. For example, you should avoid using register 12 because Language Environment uses that register when establishing the execution environment for the application.

For more information see the chapter on assembler considerations and Language Environment macros in the *OS/390 Language Environment Programming Guide*. You should also read Chapter 13, “Assembler” on page 267.

17.5 Migrating from LE/VSE

This section discusses some issues you should consider when migrating from LE/VSE-conforming languages and LE/VSE.

The items discussed here are only some of those you should consider; they are items for which the behavior in OS/390 Language Environment is different from their behavior in LE/VSE. You should also read the migration guides for OS/390 Language Environment, and for your release of LE/VSE.

17.5.1 Run-time Options

In general the run-time options with OS/390 Language Environment have the same usage and the same default values as the corresponding options in LE/VSE.

However, there are some considerations of which you should be aware when planning your migration.

The following options have different behavior in OS/390 Language Environment to their behavior in LE/VSE.

ABPERC In LE/VSE you can specify the abend code to the option ABPERC in one of three formats. These formats are:

- **S*hh***
- **I*hh***
- **U*dddd***

In OS/390 Language Environment you can only specify **S*hh*** or **U*dddd***. **I*hh*** is not allowed, and if you specify it you will receive an error message similar to:

```
CEE3616I The string 'I12' was not a valid suboption of the
run-time option ABPERC.
```

You should review your use of the ABPERC option carefully before migrating to OS/390 Language Environment as the meaning of the OS/390 system abend codes specified by **S*hhh*** are different to the VSE/ESA cancel codes specified by **S*hh***.

NATLANG The default setting for the NATLANG option in LE/VSE is NATLANG(UEN). The default setting for the NATLANG option in OS/390 Language Environment is NATLANG(ENU). UEN is upper-case U.S. English. ENU is mixed-case U.S. English.

The NATLANG option specifies the initial language to be used for the run-time environment, including error messages, month names and day-of-the-week names. If you specify an unknown national language, the error messages and so on, are displayed in the default national language.

MSGFILE The MSGFILE option has different suboptions and default values in OS/390 Language Environment to its LE/VSE counterpart. You should read the description of MSGFILE in the *OS/390 Language Environment Programming Reference*.

In LE/VSE, MSGFILE has only one suboption, *filename*. The default is SYSLST. If you specify or default to SYSLST for MSGFILE, all output from CEEMSG and CEEMOUT callable services, and RPTOPTS and RPTSTG options is written to SYSLST.

In OS/390 Language Environment there are four suboptions, *ddname, recfm, lrecl, blksize*. The defaults for these suboptions are (SYSOUT,FBA,121,0). However, if you continue to use SYSLST this is accepted by OS/390 Language Environment and the output is written to SYSOUT. If SYSOUT is specified in your OS/390 job control //SYSOUT DD SYSOUT=* then the output from CEEMSG, CEEMOUT, RPTOPTS and RPTSTG will appear in your listing.

TERMTHDACT TERMTHDACT in LE/VSE has only four suboptions, TRACE, QUIET, MSG and DUMP. TERMTHDACT in OS/390 Language Environment also has the UADUMP suboption. If specified, on an abnormal termination, the UADUMP suboption generates a system dump of the user address space. TERMTHDACT in OS/390 Language Environment is described in the *OS/390 Language Environment Programming Reference*.

Note: The UADUMP option is available in LE/VSE releases later than 1.4, and also in LE/VSE 1.4 via APAR PQ08538. Its usage is the same as for OS/390 Language Environment.

TEST The IBM defaults for the TEST option differ between LE/VSE 1.1, LE/VSE 1.4 and OS/390 Language Environment. They are:

LE/VSE 1.1 NOTEST(NONE,*,NOPROMPT,*)

LE/VSE 1.4 NOTEST(ALL,*,NOPROMPT,'')

OS/390 Language Environment
NOTEST(ALL,*,NOPROMPT,INSPPREF)

You should read the *OS/390 Language Environment Programming Reference* for information about the TEST option in OS/390 Language Environment.

Note: If you are migrating from LE/VSE 1.1, you should check your use of the TEST option carefully. In LE/VSE 1.1 the TEST option is syntax-checked only, and has no effect on the application. In OS/390 Language Environment this is not the case.

The RPTOPTS and RPTSTG options produce different reports in OS/390 Language Environment to the reports produced in LE/VSE.

RPTOPTS There are more options in OS/390 Language Environment than in LE/VSE. Therefore the options report produced by the RPTOPTS in OS/390 Language Environment will be larger than the corresponding report from LE/VSE.

RPTSTG The report produced by the RPTSTG option in OS/390 Language Environment has more information than the corresponding report produced in LE/VSE. This extra information is due to:

- Two more storage-type options in OS/390 Language Environment, NONIPTSTACK and THREADHEAP. The storage reports have information for these options.
- OS/390 Language Environment support for multithreading and multiple enclaves (see 17.1.2, “Conceptual Differences between LE/VSE and OS/390 Language Environment” on page 352). The storage report from OS/390 Language Environment has extra information for these facilities.

17.5.1.1 Run-time Options and LE/VSE 1.1

The following options were available in LE/VSE 1.1 to provide compatibility with OS/390 Language Environment. They were syntax-checked and had no effect on the application. They were removed in later releases of LE/VSE but are available in the current release of OS/390 Language Environment. If you used them in your LE/VSE 1.1 applications you should remove them or review their usage carefully. They are:

- CBLQDA
- FLOW
- INTERRUPT
- SIMVRD
- VCTRSAVE

17.5.1.2 Run-time Options and LE/VSE 1.4 and Later Releases

The following options were introduced in LE/VSE 1.4, but their usage in OS/390 Language Environment is sometimes different. They were not available in LE/VSE 1.1.

ARGPARSE This option only applies to C and can only be specified with the C `#pragma runopts` directive. `#pragma runopts` is not available with C++ so you should change your application to use the C++ **ARGPARSE** compiler option.

EXECOPS This option only applies to C and can only be specified with the C `#pragma runopts` directive. `#pragma runopts` is not available with C++ so you should change your application to use the C++ **EXECOPS** compiler option.

ENV This option only applies to C and can only be specified with the C `#pragma runopts` directive. `#pragma runopts` is not available with C++ so you should change your application to use the C++ **TARGET(IMS)** compiler option.

HEAPCHK HEAPCHK was not available in LE/VSE 1.1. It is available in LE/VSE releases later than 1.4, and in LE/VSE 1.4 via APAR PQ08538. HEAPCHK has the same behaviour in OS/390 Language Environment as in LE/VSE, and is described in *VSE/ESA Enhancements*, and in the *OS/390 Language Environment Programming Reference*.

PLIST This option only applies to C and can only be specified with the C `#pragma runopts` directive. `#pragma runopts` is not available with C++. The behavior of C applications with PLIST(HOST) in effect is the same for C++.

REDIR This option only applies to C and can only be specified with the C `#pragma runopts` directive. `#pragma runopts` is not available with C++ so you should change your application to use the C++ **REDIR** compiler option.

RETZERO The RETZERO option is a VSE-only option and not available in OS/390 Language Environment. It applies only to COBOL applications. If you are using it in your applications you should remove it. This may mean you need to make coding changes to accommodate invalid values in the RETURN-CODE special register.

If you include the RETZERO option in your OS/390 application, you will receive message:

```
CEE3611I The run-time option RETZERO was an invalid run-time option
```

Note: RETZERO is available in LE/VSE releases later than 1.4, and in LE/VSE 1.4 via APAR PQ04876.

TRACE With OS/390 Language Environment there are two more values for the sub-option LE. They are LE=2 and LE=3.

17.5.1.3 Recommended Settings for Options

The recommended settings for options for OS/390 Language Environment are described in the *OS/390 Language Environment Migration Guide*. The recommended settings for options for LE/VSE 1.4 are described in the *LE/VSE Run-Time Migration Guide Release 4*. For LE/VSE 1.1, where IBM made recommendations for the setting of options, they are described in the *LE/VSE Programming Guide Release 1*, under the description of the option.

Note: In LE/VSE 1.1, the recommended setting for most options is the default setting.

Generally, the recommendations for option settings are the same for LE/VSE and OS/390 Language Environment. Exceptions for LE/VSE 1.1 are shown in Table 42. Exceptions for LE/VSE 1.4 are shown in Table 43. Refer to the *OS/390 Language Environment Migration Guide* for further information about the recommendations for these options.

Table 42. Option Recommendations Differing between LE/VSE 1.1 and OS/390 Language Environment

Language	Option	Recommendation
COBOL	ANYHEAP	16K,8K,ANYWHERE,FREE
	BELOWHEAP	8K,4K,FREE
	DEPTHCONDLMT	10
	HEAP	32K,32K,ANYWHERE,KEEP,8K,4K
	LIBSTACK	8K,4K,FREE
	STACK	64K,64K,BELOW,KEEP
	TERMTHDACT	UADUMP
PL/I	ANYHEAP	16K,8K,ANYWHERE,FREE
	BELOWHEAP	8K,4K,FREE
	DEPTHCONDLMT	0
	HEAP	32K,32K,ANYWHERE,KEEP,8K,4K
	LIBSTACK	8K,4K,FREE
	STACK	128K,128K,BELOW,KEEP
	TERMTHDACT	TRACE

Table 43 (Page 1 of 2). Option Recommendations Differing between LE/VSE 1.4 and OS/390 Language Environment

Language	Option	Recommendation
C	ABTERMENC	ABEND
	STACK	128K,128K,BELOW,KEEP
	TERMTHDACT	TRACE
COBOL	LIBSTACK	8K,4K,FREE
	STACK	64K,64K,BELOW,KEEP
	TERMTHDACT	UADUMP

Table 43 (Page 2 of 2). Option Recommendations Differing between LE/VSE 1.4 and OS/390 Language Environment		
Language	**Option**	**Recommendation**
PL/I	ABTERMENC	RETCODE
	DEPTHCONDLMT	0
	STACK	128K,128K,BELOW,KEEP
	TERMTHDACT	TRACE

17.5.2 User Exits and Abnormal Termination Exits

This section discusses migration considerations for user and abnormal termination exits, and the similarities and differences between the exits for OS/390 Language Environment and LE/VSE.

17.5.2.1 Assembler User Exits

Three default assembler user exits are provided with LE/VSE. They are:

- CEEBXITA (batch)
- CEECXITA (CICS)
- CEEBX05A (VS COBOL II compatibility)

The default CEEBXITA is linked into the distributed batch initialization/termination phases (CEEBINIT and CEEPIPI); the default CEECXITA is linked into the distributed CICS library support routines phase, CEECCICS.

You can customize these exits to suit your requirements and link them directly to applications for use on an application-specific basis.

OS/390 Language Environment provides default assembler exits with the same names as these, and they also are linked into the batch and CICS initialization and termination load modules, and the CICS support routine load module.

However, these assembler exits may not perform exactly the same functions when invoked in OS/390 Language Environment as in LE/VSE. If you rely on the function of the default assembler exits, you should examine the OS/390 Language Environment versions to ensure they do what you require.

If you have customized your own assembler exits in LE/VSE, you can make the same customization in OS/390 Language Environment.

OS/390 Language Environment also provides a default assembler user exit for TSO, CEEBXITC, which you may find useful.

17.5.2.2 High-Level Language Exits

A sample High-Level Language (HLL) exit is provided in both LE/VSE and OS/390 Language Environment. This is CEEBINT. In both cases, it does nothing except return to the caller.

You can compile as many of your own HLL exits as you wish, in LE/VSE and in OS/390 Language Environment. In OS/390 Language Environment as in LE/VSE, you can write an HLL exit in C, PL/I or Language Environment-conforming Assembler language, but not in COBOL.

Note: You cannot write an HLL exit in C, for use in LE/VSE 1.1.

In OS/390 Language Environment a sample job, CEEWHLLX, is provided that contains an SMP/E USERMOD to replace the IBM-supplied HLL user exit with your HLL user exit.

17.5.2.3 Abnormal Termination Exits

Language Environment provides the ability to invoke an abnormal termination exit before it terminates a thread due to an unhandled condition of severity 2 or greater. This allows an abnormal termination exit to collect problem determination data before Language Environment frees the resources it has acquired.

Abnormal termination exits can be invoked in CICS or non-CICS. You can code your own abnormal termination exits. In LE/VSE this is described in the *LE/VSE Installation and Customization Guide*. In OS/390 Language Environment this is described in *OS/390 Language Environment Customization*.

In OS/390 Language Environment (as with LE/VSE 1.1) no default or sample abnormal termination exits are supplied.

17.5.2.4 Abnormal Termination Exits and LE/VSE 1.4 and Later Releases

In LE/VSE 1.4 and later releases, default abnormal termination exits are supplied. These are CEEBDATX for batch and CEECDATX for CICS. CEEBDATX is a null module that immediately returns to the caller when invoked. CEECDATX issues abend 4039 when an unhandled condition occurs of severity 2 or greater.

In LE/VSE 1.4 and later releases, sample source programs are supplied that can be used as examples of how to write an abnormal termination exit. These are CEEBBATX.A and CEEBNATX.A. CEEBBATX.A is a batch abnormal termination exit that produces a system dump when invoked. CEEBNATX.A is a batch or CICS exit that does nothing but return to the caller when invoked.

17.5.3 Callable Services and Math Services

All LE/VSE callable services and math services are also provided in OS/390. The use of callable and math services in OS/390 Language Environment is described in the *OS/390 Language Environment Programming Reference*.

If you use LE/VSE callable services, there is one important fact of which you should be aware. Some LE/VSE callable services have names beginning with **CEE5**. The corresponding OS/390 Language Environment callable services have names beginning with **CEE3**. For example, the callable service in LE/VSE, to set the heading displayed at the top of the options report, is **CEE5RPH**. The corresponding OS/390 Language Environment callable service is **CEE3RPH**.

There are 14 such callable services. They are listed in Figure 56 on page 366. If you have used any of these callable services in your programs, you must change their names in your source code when you transfer the code to OS/390, and you must recompile your source code there. You cannot use the OS/390 Language Environment names in your VSE/ESA code, you cannot use the LE/VSE names in your OS/390 Language Environment code, and you cannot ship the compiled object code from VSE/ESA to OS/390 for link-editing there.

CEE5ABD	CEE5GRN	CEE5MDS	CEE5SPM
CEE5CIB	CEE5GRO	CEE5MTS	CEE5SRC
CEE5CTY	CEE5LNG	CEE5PRM	CEE5SRP
CEE5DMP	CEE5MCS	CEE5RPH	CEE5USR
CEE5GRC			

Figure 56. Callable Services in LE/VSE 1.4 with Differing Names in OS/390 Language Environment

Note: Three further callable services are available in LE/VSE releases later than 1.4, and in LE/VSE 1.4 via APAR PQ08538. They are also available in OS/390 Language Environment. They are:

- CEEMRCE
- CEE4SRP
- CEE5GRO

17.5.3.1 CEETDLI

The CEETDLI callable service in LE/VSE provides an interface to DL/I DOS/VS facilities. The OS/390 Language Environment callable service CEETDLI provides an interface to DL/I facilities that operate in IBM and CICS. You should read Chapter 8, “Databases” on page 169 and the appropriate DL/I and IMS publications, for more information.

17.5.4 LE/VSE 1.4 Locales

All locales provided in LE/VSE 1.4 are also provided in OS/390. This includes a number of locales and charmaps available in LE/VSE releases later than 1.4, and in LE/VSE 1.4 via APARs PQ08543 and PQ08547. Locales and the localedef utility, for OS/390 Language Environment, are described in the *OS/390 C/C++ V2R4.0 Programming Guide*.

17.6 CICS

This section discusses migration issues relating specifically to CICS.

17.6.1 COBOL and CICS

In OS/390 Language Environment, as in LE/VSE, some Language Environment COBOL run-time routines have the same names as their non-CICS counterparts. Therefore, if you plan to run COBOL programs in CICS, you must concatenate the library containing the Language Environment COBOL run-time routines in front of the library containing non-CICS routines, in the DFHRPL DD concatenation in your CICS startup job stream. Generally, the name of the Language Environment COBOL library is SCEECICS and the name of the non-CICS library is SCEERUN.

17.6.2 Run-time Options

The default settings for run-time options, for CICS, are the same for OS/390 Language Environment and for LE/VSE. Refer to the *OS/390 Language Environment Programming Reference* for the OS/390 Language Environment default settings.

The recommended settings for run-time options for CICS, are the same for OS/390 Language Environment and for LE/VSE, with the following exceptions, listed in Table 44 on page 367.

Table 44. Option Recommendations for CICS Differing between LE/VSE and OS/390 Language Environment

Language	Option	Recommendation
COBOL	LIBSTACK	1K,1K,FREE
	TERMTHDACT	UADUMP
PL/I	DEPTHCONDLMT	0
	ERRCOUNT	0

17.6.3 User Exits and Abnormal Termination Exits

These are discussed in 17.5.2, “User Exits and Abnormal Termination Exits” on page 364.

Chapter 18. Procedure Language REXX

The REstructured eXtended eXecutor language, or REXX language, is a versatile, easy to use structured procedure language that is part of:

- VM/ESA
- VSE/ESA
- TSO/E

REXX was designed as a replacement for the EXEC and EXEC2 languages that provided a way to bundle Conversational Monitor System (CMS) commands together. REXX is an extremely versatile programming language in that it can be intermixed with commands to host environments, it provides powerful functions, and it has extensive mathematical capabilities.

18.1 REXX and VM/ESA

By far, the most vital role REXX plays is as a procedural language for VM/ESA. That means a REXX procedure can be a kind of script for VM/ESA to follow. By using REXX, you can reduce long and complex or repetitious tasks to a single command or procedure that can be run from CMS.

REXX is a built-in feature of VM/ESA, so there is no installation process or separate environment. Any REXX procedure can call CMS commands, XEDIT macros, other subcommand environments, GCS and so on.

18.2 REXX and VSE/ESA

REXX programs can do many tasks, including the automation of VSE/Operations. For example, if you use the JCL EXEC command to call a REXX program, you can leave JCL statements on the stack for VSE/ESA to process. This enables you to insert JCL statements or data into the current job stream. REXX programs can run in any partition. They can communicate with POWER through the Spool Access Support interface.

REXX/VSE is available for all VSE/ESA 2.1 and higher and is integrated closely into the VSE/ESA central functions. It is implemented through:

- The REXX/VSE interpreter
- The Library for REXX/370 in REXX/VSE

18.3 REXX and TSO/E

The TSO/E implementation of the REXX language allows REXX execs to run in any MVS address space. You can write a REXX exec that includes TSO/E services and run it in a TSO/E address space, or you can write an application in REXX to run outside of a TSO/E address space. REXX runs in an OS/390 system in different environments: MVS, NetView, OE shell scripts, ...

18.4 Environments

The system under which REXX procedures run is assumed to include at least one environment for processing commands. An environment is selected by default on entry to a REXX procedure.

You can change the environment by using the ADDRESS instruction:

```
address cms        /* VM/ESA CMS  environment  */
address POWER      /* VSE/ESA POWER host command environment  */
address TSO "ALLOC F(SYSOUT) DSN(my.dsn) SHR"
address DOS 'DIR MDV1.ALL'
```

TSO ISPF dialog invocation:

```
address ISPEXEC "SELECT CMD(%myexec) "
```

You can find out the name of the current environment by using the ADDRESS built-in function. ADDRESS() returns the name of the environment to which commands are currently being submitted.

```
if address() = 'CMD' then   /* OS/2 environment           */
```

The underlying operating system defines environments external to the REXX procedure.

The environment selected depends on the caller. For example if a procedure is called from CMS, the default environment is CMS. If called from an editor that accepts subcommands from the language processor, the default environment may be that editor.

ADDRESS temporarily or permanently changes the destination of commands. Commands are strings, not interpreted by REXX itself but sent to an external environment.

18.4.1 VSE/ESA Environment

REXX/VSE provides the following host command environments:

- VSE for the REXX/VSE commands.
- POWER for VSE/POWER spool-access service requests.
- JCL to issue JCL commands via a REXX program.
- CONSOLE lets you manage console sessions.
- LINK and LINKPGM for linking to a program.

18.4.2 VM/ESA Environment

VM/ESA provides among others the following host command environments:

- CMS implies full command resolution just as provided in usual interactive command (terminal) mode.
- COMMAND implies basic CMS CMSCALL command resolution.
- CPICOMM can be used to call program-to-program communications routines.
- OPENVM can be used to call OPENVM-type CSL routines, such as OpenEdition for VM/ESA callable services.

18.4.3 TSO/E Environment

TSO/E provides among others the following host command environments:

- TSO allows you to invoke TSO/E commands and services.
- CONSOLE allows you to invoke MVS system and subsystem commands during an extended MCS console session.
- ISPEXEC and ISAREDIT allows you to invoke ISPF commands and services, and ISPF edit macros.
- CPICOMM, LU62, and APPCMVS allows you to use the SAA common programming interface (CPI) Communications calls.
- MVS gives you a host environment which is available in any MVS address space.

18.4.4 REXX Exec Sample for the OS/2, TSO and CMS Environments

```
/*   REXX    */
say 'REXX Exec is executed in the' address() 'environment'
if address() = 'CMD' then           /* OS/2 environment             */
  do
   infile = 'my.data'
   do until lines(infile) = 0
      say linein(infile)
   end
  end
else
  do
   if address() = 'TSO'  address() = 'CMS' then
     do
      io_op = "EXECIO * DISKR"      /* common used EXECIO part      */
      if address() = 'TSO' then     /* TSO environment              */
       do
        e1 = "ALLOC FI(DATAIN) DA(my.data) SHR"
        e2 = io_op "DATAIN (FINIS"
       end
      if address() = 'CMS' then     /* CMS environment              */
       do
        e1 = io_op "my data A"
        e2 = "FINIS my data A"
       end
      e1;e2                         /* execute I/O                  */
      do i = 1 to queued()
       parse pull line
       say line
      end
     end
  end
```

18.5 Migration Issues

"The REXX language is independent of both system and hardware. REXX procedures, though, must be able to interact with their environment. Such interactions necessarily have system dependent attributes. However, these system dependencies are clearly bounded and the rest of the language has no such dependencies." (M. F. Cowlishaw: The REXX Language)

REXX is compatible with the VM, VSE, MVS, AIX, OS/400, and OS/2 operating systems, among others, and allows system independent coding.

18.5.1 REXX and SAA

Issuing commands to the surrounding environment is an integral part of REXX. REXX is the only procedure language supported by the SAA to help provide cross-system consistency. Procedures written in REXX according to the SAA specifications can be transported to other SAA environments. For example, a REXX exec in CMS can also run in a TSO/E environment if the exec does not use system-specific functions or commands.

Only the ADDRESS command/instruction affects the host command environment of the exec that uses the command/instruction.

18.6 REXX Bibliography

VSE/ESA

VSE/REXX Reference, SC33-6642
VSE/REXX User's Guide, SC33-6641
VSE/REXX Console Automation, SC33-6598

VM/ESA

VM/ESA REXX/VM User's Guide, SC24-5465
VM/ESA REXX/VM Reference, SC24-5770

TSO/E

OS/390 V1R2.0 TSO/E REXX Reference, ST01-2613
OS/390 OPENEDITION MVS using REXX and OPENEDITION MVS, ST01-2695

Part 4. Converting VSE Utilities to OS/390 Utilities

In addition to this part of the book on converting utilities, also see Chapter 29, “Orientation for Utilities” on page 455 which discusses more OS/390 Utilities that you can use.

Chapter 19. SORT

This chapter addresses considerations for migrating to the OS/390 Sort product, DFSORT (5740-SM1) from the VSE Sort products:

- DOS/VS SORT/MERGE V2 (5746-SM2), referred to as Sort/Merge
- DFSORT/VSE V3 (5746-SM3), referred to as DFSORT/VSE

DFSORT/VSE is based on and replaces Sort/Merge. It offers additional features not available with Sort/Merge. The migration considerations for Sort/Merge will be discussed first. These considerations also apply to DFSORT/VSE. Migration considerations for additional features of DFSORT/VSE are discussed separately at the end of the chapter.

The interfaces for calling DFSORT from a program and for DFSORT user exit routines are significantly different from the corresponding interfaces for Sort/Merge. Your calling programs and user exits thus require careful consideration and redesign, as appropriate. These interfaces are described fully in the *DFSORT Application Programming Guide*, SC33-4035 and will not be discussed here.

Sort/Merge, DFSORT/VSE and DFSORT were designed to be functionally compatible at the control statement level, and for the most part, the control statement syntax of the three products is compatible. However, differences in JCL, data set usage and control statement syntax must be addressed when migrating to DFSORT. These differences are summarized in this chapter.

DFSORT has many features that are not available with Sort/Merge and some features that are not available with DFSORT/VSE. These additional DFSORT features are only discussed here when they relate to migration considerations. However, you may want to familiarize yourself with these additional features since they can be useful for your DFSORT applications.

For complete details on DFSORT, see the *DFSORT Application Programming Guide*, SC33-4035.

The following topics are discussed:

- 19.1, JCL Statements
- 19.2, Control Statements
- 19.3, Additional DFSORT/VSE Migration Considerations

19.1 JCL Statements

As discussed in Chapter 4, “Job Control Language (JCL) Differences and Considerations” on page 69, the JCL for OS/390 is quite different from that for VSE. You will need to replace your VSE JCL statements with appropriate OS/390 JCL statements. Every DFSORT job requires a JOB statement and an EXEC statement. DFSORT jobs also require specific DD statements which depend on the type of application being run (sort, merge or copy) as well as specific features you want to use for the job.

A basic DFSORT job might look as follows:

```
//MYJOB JOB ...
//SORTIT EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=...
//SORTOUT DD DSN=...
//SYSIN DD *
  SORT FIELDS=(5,4,CH,A)
/*
```

The JCL statements you will commonly need when migrating are:

JOB: Must be the first statement for a job.

EXEC: Must be the first statement for a step. Specifies the program to be executed. For DFSORT steps, use PGM=SORT or PGM=ICEMAN.

SYSOUT DD: Defines DFSORT′s message data set. DFSORT can write informational, error and diagnostic messages to this data set as directed.

SORTIN DD: Defines one or more input data sets for a sort or copy application. Multiple input data sets can be specified using OS/390′s data set concatenation facility.

SORTINnn DD: Defines a data set for a merge application. SORTIN01 through SORTIN99 can each be used to specify a data set to be merged.

SORTOUT DD: Defines the output data set for a sort, merge or copy application.

SYSIN DD: Contains DFSORT control statements, comment statements and blank statements.

Other JCL statements you may need when migrating are:

JOBLIB DD: Defines the library containing the DFSORT program. Used for all steps in the job. The JOBLIB DD is only needed if the DFSORT library is not known to the operating system.

STEPLIB DD: Defines the library containing the DFSORT program. Used for a particular step in the job. The STEPLIB DD is only needed if the DFSORT library is not known to the operating system.

SORTWKnn DD: Defines a work data set for a sort application. It is recommended that you use DFSORT′s dynamic allocation facility instead of specifying SORTWKnn DD statements. DFSORT′s shipped defaults result in the automatic use of dynamic allocation when appropriate.

SORTDIAG DD: Forces DFSORT to print all messages and control statements in the message data set. Only needed for diagnosing problems.

SYSUDUMP DD: Defines the data set for a dump. Only needed for diagnosing problems.

19.2 Control Statements

DFSORT was designed to be functionally compatible with Sort/Merge at the control statement level, and for the most part, the control statement syntax of the two products is compatible. However, there are differences in some of the control statements that you will need to address. Here are the actions, if any, you should consider taking for each Sort/Merge control statement.

ALTSEQ: Can be used with no changes.

ANALYZE: Must be removed. DFSORT terminates if an ANALYZE statement is specified. Use DFSORT's dynamic allocation feature to allow DFSORT to automatically use the work space needed. Use SORTDIAG DD to produce diagnostic messages.

END: Can be used with no changes.

INCLUDE: Can be used with no changes.

INPFIL: Must be removed if the INPFIL statement is continued because DFSORT control statement errors can result. If the INPFIL statement is not continued, it can be used with no changes. DFSORT ignores all INPFIL operands. The equivalent information must be available from the input DD statements, input data set control blocks (DSCB) or catalog.

INREC: Can be used with no changes.

MERGE: Can be used with no changes.

MODS: Must be changed to use DFSORT syntax. User exit routines must be changed to use the DFSORT interfaces. See the *DFSORT Application Programming Guide*, SC33-4035 for complete details on the MODS statement and the interfaces for user exit routines.

OMIT: Can be used with no changes.

OPTION: Here are the actions, if any, you should consider taking for each Sort/Merge OPTION operand:

- **ADDROUT:** Must be removed. DFSORT terminates if this operand is specified. DFSORT does not support the use of direct-access addresses in output records. If you want to continue to include such addresses, you must write an E15 user exit to handle this.
- **CHALT:** Can be used with no changes.
- **NOCHALT:** Can be used with no changes.
- **DIAG:** Can be used with no changes. DFSORT ignores DIAG. Use the SORTDIAG DD statement to obtain diagnostic messages.
- **NODIAG:** Can be used with no changes. DFSORT ignores NODIAG.
- **DUMP:** Must be removed. DFSORT terminates if this operand is specified. Specify a SYSUDUMP DD statement to obtain a dump.
- **NODUMP:** Must be removed. DFSORT terminates if this operand is specified. Do not specify a SYSUDUMP DD statement to suppress a dump.

- **ERASE:** Can be used with no changes. DFSORT ignores ERASE. Use a security product, such as RACF, to erase the work data sets.
- **NOERASE:** Can be used with no changes. DFSORT ignores NOERASE. DFSORT does not erase work data sets.
- **FILNM:** Must be removed. DFSORT terminates if this operand is specified. Use DFSORT′s SORTIN, SORTOUT or SORTDD operands to change the input and output ddnames.
- **LABEL:** Must be removed. DFSORT terminates if this operand is specified. Use the LABEL option of the DD statement to specify the type of label.
- **PRINT:** Must be removed. DFSORT terminates if this operand is specified. If the default message level is inappropriate for a particular job, use DFSORT′s MSGPRT operand to control which messages are printed.
- **ROUTE:** Must be removed. DFSORT terminates if this operand is specified. By default, DFSORT directs its messages to the message data set. However, DFSORT′s MSGCON installation operand can be used to direct messages to the master console.
- **SORTIN:** Must be removed. DFSORT terminates if this operand is specified. Use DD statements to identify the input data sets.
- **SORTOUT:** Must be removed. DFSORT terminates if this operand is specified. Use a DD statement to identify the output data set.
- **SORTWK:** Must be removed. DFSORT terminates if this operand is specified. Use DFSORT′s DYNALLOC operand or SORTWKdd DD statements to identify the work data sets.
- **STORAGE:** Must be removed. DFSORT terminates if this operand is specified. By default, DFSORT uses virtual storage (above and below 16MB virtual), dataspace sorting, hipersorting and work data sets, as appropriate. If the default storage values at your site are inappropriate for a particular job, use DFSORT′s MAINSIZE operand to control storage.
- **VERIFY:** Is accepted, but performs a different function for DFSORT than for Sort/Merge. Use the OPTCD=W option on the SORTOUT DD statement to perform the equivalent of Sort/Merge′s VERIFY function.
- **NOVERIFY:** Can be used with no changes. DFSORT will not perform its VERIFY function.
- **WORKNM:** Must be removed. DFSORT terminates if this operand is specified. Use DFSORT′s SORTDD operand to change the work ddnames.

OUTFIL: Can be used with no changes. DFSORT ignores all Sort/Merge OUTFIL operands. The equivalent information must be available from the output DD statement, output data set control block (DSCB) or catalog.

OUTREC: Can be used with no changes.

RECORD: Can be used with no changes.

SORT: Can be used with no changes. DFSORT ignores operands that are not meaningful for its processing.

SUM: Can be used with no changes.

19.3 Additional DFSORT/VSE Migration Considerations

DFSORT/VSE is based on and replaces Sort/Merge. It offers additional features not available in Sort/Merge. All of the migration considerations discussed previously for Sort/Merge apply to DFSORT/VSE as well. Migration considerations for additional features of DFSORT/VSE are discussed below.

19.3.1 Control Statements

OPTION: DFSORT/VSE has additional OPTION statement operands not found in Sort/Merge. Here are the actions, if any, you should consider taking for each such OPTION operand:

- **DSPSIZE:** Can be used with no changes. However, since the OS/390 environment is considerably different from the VSE environment, you should determine if the DSPSIZE value specified is still appropriate as the maximum amount of data space to be used for dataspace sorting.
- **EQUALS:** Can be used with no changes.
- **NOEQUALS:** Can be used with no changes.
- **GVSIZE:** Must be removed. DFSORT terminates if this operand is specified. GVSIZE has no meaning for DFSORT since OS/390 does not support GETVIS areas. By default, DFSORT uses virtual storage (above and below 16M virtual), dataspace sorting, hipersorting and work data sets, as appropriate.
- **GVSRANY:** Must be removed. DFSORT terminates if this operand is specified. GVSRANY has no meaning for DFSORT since OS/390 does not support GETVIS areas.
- **GVSRLOW:** Must be removed. DFSORT terminates if this operand is specified. GVSRLOW has no meaning for DFSORT since OS/390 does not support GETVIS areas.
- **LOCALE:** Can be used with no changes.
- **SKIPREC:** Can be used with no changes.
- **STOPAFT:** Can be used with no changes.
- **STXIT:** Must be removed. DFSORT terminates if this operand is specified. By default, DFSORT uses an ESTAE routine for abend recovery.
- **NOSTXIT:** Must be removed. DFSORT terminates if this operand is specified. Use the NOESTAE operand of the DEBUG control statement to suppress DFSORT′s ESTAE routine for abend recovery.
- **WRKSEC:** Must be removed. DFSORT terminates if this operand is specified. By default, DFSORT uses automatic secondary allocation for temporary JCL SORTWKnn data sets for which a secondary allocation amount is not specified.
- **NOWRKSEC:** Can be used with no changes. However, since NOWRKSEC applies to SAM ESDS work files for Sort/Merge, but to temporary work files for DFSORT, you should determine if this operand should be kept or removed.
- **Y2PAST:** Can be used with no changes.

OUTREC: DFSORT/VSE has additional OUTREC statement operands not found in Sort/Merge. Here are the actions, if any, you should consider taking for each such OUTREC operand:

- **p,m,Y2x:** Must be removed. DFSORT terminates if this operand is specified. p,m,Y2x can be specified in the OUTREC operand of DFSORT′s OUTFIL statement.
- **p,m,PZ:** Must be removed. DFSORT terminates if this operand is specified. p,m,PZ can be replaced by p,m,PD0,M11 in the OUTREC operand of DFSORT′s OUTFIL statement.
- **p,m,PSI:** Must be removed. DFSORT terminates if this operand is specified. p,m,PSI can be replaced by p,m,PD,M11 in the OUTREC operand of DFSORT′s OUTFIL statement.
- **p,m,ZSI:** Must be removed. DFSORT terminates if this operand is specified. p,m,ZSI can be replaced by p,m,ZD,M11 in the OUTREC operand of DFSORT′s OUTFIL statement.

19.3.2 ICETOOL

DFSORT/VSE′s ICETOOL and DFSORT′s ICETOOL were designed to be functionally and syntactically compatible. However, differences between the VSE and OS/390 operating systems require the following changes when migrating:

- **JCL Statements:** The VSE JCL statements describing the files used by ICETOOL must be changed to DD statements as described in the *DFSORT Application Programming Guide*, SC33-4035.
- **DEFINE operator:** Must be removed. ICETOOL terminates if this operator is specified. ICETOOL obtains information about each input data set from the DD statement, data set control block (DSCB) or catalog.
- **FROM operand:** Each FROM operand that specifies more than one filename must be changed to specify one ddname that identifies the input data sets using concatenation.
- **LIST operand:** Each LIST operand must be changed to specify a ddname that identifies the list data set.
- **USE operand:** Each USE operand must be replaced by a USING(xxxx) operand where xxxx is a unique four-character identifier. The control statements between (but not including) the USTART and UEND statements must be placed in a data set defined by an appropriate xxxxCNTL DD statement.

Chapter 20. DITTO

Data Interfile Transfer, Testing, and Operations Utility (DITTO) is IBM′s best known storage media and data maintenance utility program for the OS/390, MVS, VSE, and VM environments.

DITTO is a key resource for data processing professionals due to many versatile functions working with tapes, disks, VTOCs and catalogs, VSAM data, VSE library members, sequential data sets and files, MVS Object Access Method (OAM) objects, and card images.

DITTO/ESA introduced an all in one User′s Guide to let users of OS/390, MVS/ESA, VSE/ESA, and VM/ESA compare the functions and parameters supported in each one of these S/390 environments.

DITTO/ESA Release 2 is the latest release of the well known DITTO family of products. It supersedes DITTO/ESA Release 1 (which superseded MVS/DITTO Version 2.1, the DITTO 3.2 Productivity Features for VSE and VM, and DITTO for VSE and VM 3.2 base product).

It provides a consistent package of functions for the MVS, VSE, and VM user with a common user interface supporting the following operating system environments:

- OS/390
- MVS/ESA
- VSE/ESA
- VM/ESA

The following topics will be discussed:

- 20.1, Compatibility with Previous Releases of DITTO
- 20.2, DITTO Functions that are No Longer Supported
- 20.3, DITTO Functions that are Not Recommended
- 20.4, DITTO Function Code Synonyms
- 20.5, Batch Keywords that are No Longer Supported
- 20.6, Batch Keywords that are Not Recommended
- 20.7, DITTO/ESA Security

20.1 Compatibility with Previous Releases of DITTO

This section describes the differences between DITTO/ESA and previous versions of DITTO:

- MVS/DITTO Version 2 Release 1 (Program Number 5695-100)
- DITTO Version 3 Release 2 for VSE and VM (Program Number 5688-052)
- DITTO Version 3 Release 2 Productivity Features for VSE and VM

Under VSE and CMS, the DVT and VDL functions have been changed. Previously, an asterisk (*) within a file ID represented any number of characters. Now, one asterisk represents any number of characters within a qualifier and a

double asterisk (**) represents any number of characters in any number of qualifiers.

Under VSE and CMS, an implicit rewind was previously performed by each function that works with labeled tapes. Labeled tape functions could only work with the first file on a tape. The implicit rewind is no longer performed. This lets you work with multifile standard labeled tapes.

20.2 DITTO Functions that are No Longer Supported

The following table lists function codes that were allowed in previous releases of DITTO but are not recognized by DITTO/ESA. You can use the indicated replacement, if any.

Function	Description	Replacement
BDU	Buffer to Diskette	-
BIS, BI	Buffer to ISAM	-
CDU	Card to Diskette	-
DDD	Disk Dump with Deblocking	DP
DPD	Disk to Printer with Deblocking	DP
DKB	Diskette Browse	-
DKE	Diskette Eject and Feed	-
DKH, DKP	Diskette to Printer	-
DKI	Display Diskette Index	-
DKL	Diskette Record Load	-
DKN	Diskette to Console	-
DKS	Diskette Record Scan	-
DKV	Diskette Identification Change	-
DUC, DUI	Diskette File to Card	-
DUD	Diskette File to Diskette File	-
DUF	Diskette File to CMS File	-
DUH, DUP	Diskette File to Printer	-
DUS	Diskette File to SAM File	-
DUT	Diskette File to Tape	-
DUV	Diskette File to VSAM	-
FDD	CMS File Dump Deblocked	FP
FDU	CMS File to Diskette File	-
IDK	Initialize Diskette	-
IDP, ID	ISAM File Dump	-
IIS	ISAM File to ISAM File	-
IPR, IP	ISAM File Print	-
ISQ, IS	ISAM File to SAM File	-
ITP, IT	ISAM File to Tape	-
IVS, IV	ISAM File to VSAM	-
LE	Library Member Erase	LDEL

Function	Description	Replacement
SD, SDD	Split-Cylinder Disk Dump	-
SDU	SAM to Diskette	-
SIS	SAM File to ISAM File	-
SP, SPD	Split-Cylinder Disk Print	-
SRS	Split-Cylinder Disk Record Scan	-
TDD	Tape Dump Deblocked	TP
TDU	Tape to Diskette	-
TDV	Tape Dump Variable	TP
TIS	Tape to ISAM File	-
TPD	Tape Print Deblocked	TP
TPV	Tape Print Variable	TP
VDU	VSAM to Diskette File	-
VIS	VSAM to ISAM File	-

20.3 DITTO Functions that are Not Recommended

The following table lists obsolete function codes from previous releases of DITTO that are still recognized by DITTO/ESA in batch mode. It is recommended that you use the indicated replacement.

Function	Description	Replacement
CD	Card Dump	CP
CI	Card to Card Interpreted	CC
DD	Disk Dump	DP
FD	CMS File Dump	FP
LD	Library Member Dump	LP
LI	Library Member to Card Interpreted	LC
OD	Object Dump	OP
QD, SDP	Sequential Data Dump	SP
QI, SI	Sequential Data to Card Interpreted	SC
TD	Tape Dump	TP
TI	Tape to Card Interpreted	TC
VD, VDP	VSAM Dump	VP
VI	VSAM to Card Interpreted	VC

20.4 DITTO Function Code Synonyms

The following table lists supported synonyms for DITTO function codes.

Function	Synonym(s)	Description
BS	BQ, BSQ	Create Sequential Data
BT	BTP	Create Tape File
BV	BVS	Create VSAM File
CS	CQ, CSQ	Card to Sequential Data
CV	CVS	Card to VSAM
DUMP	DUM	Dump CMS File to Tape
FP	FPR	CMS File Print
FT	FTP	CMS File to Tape
FV	FVS	CMS File to VSAM
LOAD	LOA	Load CMS File from Tape
MB	TST	Memory Browse
NEW	NEWS	News Command
OS	OQ, OTL	Object to Sequential Data
SC	QC	Sequential Data to Card
SO	QO, TLO	Sequential Data to Object
SP	QP, SPR	Sequential Data Print
SS	QQ, SSQ	Sequential Data to Sequential Data
ST	QT, STP	Sequential Data to Tape
SV	QV, SVS	Sequential Data to VSAM
TS	TQ, TSQ	Tape to Sequential Data
TV	TVS	Tape to VSAM
VP	VPR	VSAM Print
VRU	VRL	VSAM Record Update
VS	VQ, VSQ	VSAM to Sequential Data
VT	VTP	VSAM to Tape
VV	VVS	VSAM to VSAM

20.5 Batch Keywords that are No Longer Supported

The following table lists keywords that were allowed in previous releases of DITTO but are not recognized by DITTO/ESA. You can use the indicated replacement, if any.

Function(s)	Keyword	Description	Replacement
BV	BLKFACTOR=	Output blocking factor	-
TMP	TYPE=CHAR	Tape map in character format	FORMAT=CHAR

20.6 Batch Keywords that are Not Recommended

The following table lists obsolete keywords from previous releases of DITTO that are still recognized by DITTO/ESA in batch mode. It is recommended that you use the indicated replacement, if any.

Function(s)	Keyword	Description	Replacement
BS, BT, CS, CT, FT, SS, ST, TFT, TS, TTR, VS, VT	BLKFACTOR=	Output blocking factor	RECFMOUT= BLKSIZE=
TF	BLKFACTOR=	Output blocking factor	-
BS, CS, SS, TS, VS	CISIZE=	FBA control interval size	BLKSIZE=
FT	MODE=	Input record format	-
SP (VSE, CMS), SS, ST (VSE, CMS), SV, TC, TF, TFT, TRS, TS, TTR, TV	MODE=	Input record format	RECFMIN=
BT, CT, ST (MVS), VS, VT	MODE=	Output record format	RECFMOUT=
TTR	OUTMODE=	Output record format	RECFMOUT=
TP, TRS	NBLKS=	Number of blocks	NLRECS=
FT	RECSIZE=	Record length for deblocking	-
VC, VL, VP, VRU, VS, VT, VV	START=K POSITION=key	VSAM start positioning by key value	KEY=key
SET	TSOPRINT=	Print output destination	PRINTOUT=
SET	ASCII=YES	Translate from ASCII to EBCDIC	ASCII=IN

Notes:

1. For the SP function, the MODE keyword is obsolete under VSE and CMS but still applies under MVS.
2. For the ST function, the MODE keyword is obsolete. Under VSE and CMS, use RECFMIN; under MVS, use RECFMOUT.

20.7 DITTO/ESA Security

DITTO provides secure control of function authorization, either through RACF (or an equivalent security product) or through the DITSECUR exit.

DITSECUR is a customizable exit. It provides a DITS macro, which lets you define a table of user names or job names, DITTO-protectable resources (called profiles), and access levels.

OS/390, MVS, or CMS: If OS/390 Security Server, RACF 1.9 or later (or equivalent) is active, the System Authorization Facility (SAF) with the DITTO enhanced security facility is used for access control and authorization verification. Authorization is controlled by DITTO-specific profiles in the FACILITY class. If SAF with RACF 1.9 is not active at DITTO initialization time, all DITTO special security checks during that DITTO session are passed to the DITSECUR user exit (if any) instead of to SAF; if the DITSECUR module cannot be found, no security checks are done.

VSE: All DITTO special security checks during a DITTO session are passed to the DITSECUR user exit. If the DITSECUR module cannot be found, no security checks are done.

Chapter 21. VSAM Backup/Restore

21.1 VSAM Backup/Restore

The following describes the methods and utilities used in OS/390 as compared to VSE VSAM Backup/Restore procedures.

21.1.1 OS/390 VSAM Backup/Restore

There are several tools that can be used in an OS/390 system to back up and restore VSAM files. Considerations for which tool you may want to use can be found in Chapter 4 of *DFSMS/MVS Using Data Sets*, SC26-4922. A brief description of these tools follows:

- DFSMSdfp

  DFSMSdfp provides the IDCAMS commands, EXPORT/IMPORT, for backup and restore of VSAM files. These commands are documented in the *DFSMS/MVS Access Methods Services for ICF* SC26-4906.

- DFSMShsm

  DFSMShsm can be used to back up and restore VSAM files. For more information see *DFSMShsm Managing Your Own Data*. In addition, backup of data can be done automatically with DFSMShsm in a non-SMS or SMS environment.

- DFSMSdss

  DFSMSdss can be used for VSAM backup/restore. See the manual *DFSMSdss Storage Administration Reference*, SC26-4929, for command syntax as well as hints and tips.

- RVA/SnapShot V1R2

  With RVA and SnapShot V1R2, you can SNAP a VSAM data set to disk and offload it to tape at a later time.

21.1.2 VSE/VSAM Backup/Restore

You can use the VSE/VSAM Backup/Restore feature to back up VSE/VSAM files to magnetic tape or disk devices, and restore the files again to VSE/VSAM data sets. You can use the two IDCAMS commands BACKUP and RESTORE.

Use this feature to write and read data sets as follows:

- Copy VSAM disk files to magnetic tape or disk (BACKUP).
- Copy from magnetic tape or disk to VSAM disk files (RESTORE).

The operations listed below can be done for the following VSE/VSAM objects:

- KSDS files
- ESDS files
- RRDS files
- VRDS files
- Alternate indexes
- SAM ESDS files in CI format

Operations:

1. Handle several VSE/VSAM files with a single command, either with a generic name or as files of one catalog.
2. Restore VSE/VSAM files to locations, volumes, and device types that are different from those where the files were before.
3. Exclude files from a collective backup or restore operation.
4. Tune the performance of VSE/VSAM by specifying the size of the buffer in the BACKUP command, and the number of buffers in both the BACKUP and RESTORE commands.

In addition, the VSE/VSAM Backup/Restore Feature allows you to:

- Back up and restore empty objects, where an empty object may be either a:
  - VSE/VSAM object defined with NOALLOCATION (such as a default model or a dynamic file), or
  - VSE/VSAM cluster that has not been loaded since being defined or reset.
- Change the allocation size for the data component of a file at restoration. You can specify allocation size in device-independent units by using the RECORDS parameter when the cluster is defined to facilitate restoration of objects.
- Change the index CI-size at restoration.

Chapter 22. Librarian

Both VSE and OS/390 have facilities to help you define, organize, and manage libraries of system data. In VSE these system facilities are called the *Librarian*.

In OS/390, the equivalent function is provided by Partitioned Data Sets (PDS) and the newer Partitioned Data Set-Extended (PDSE), and the group of utilities used for their management, including the Partitioned Access Method (PAM) and the Interactive System Productivity Facility (ISPF).

The following briefly describes VSE and OS/390 library support.

22.1 Overall Library Support

One of the most important utilities in VSE is the VSE Librarian which allows you to back up, restore and re-organize VSE libraries. In OS/390, there are a group of utilities which provide similar functions and capabilities.

Generally, a librarian is a program or a group of programs which serve to organize and maintain the libraries of a system including both system and user program source, object, and load modules. It also contains service functions to display and punch parts of them or display their directories and to set up and change the library concatenation chains and their control tables.

The concept of a librarian is based on the following aspects:

- ***Logical library structure***

  A VSE library is logically divided into sublibraries. Each sublibrary may contain members of any type (procedures, source books, object modules, phases, user-defined types). This makes a library common for all types of members. Therefore, a component consisting of procedures, source books, object modules, and phases may be put into one library and even into one sublibrary.

  In OS/390, a partitioned data set contains a directory and a collection of members. A single PDS (or PDSE) logically corresponds to a sublibrary in the VSE paradigm. It is usual, however, to place members of different types (source, object or load members) into different PDSs in OS/390 environments.

- ***Library data format***

  VSE library data is blocked, resulting in good utilization of DASD space. The VSE library data format allows space to be reused. Space freed due to deletion of a member is available for reuse without requiring a condense run. The member directories are sorted, improving retrieval times.

  Partitioned Data Sets in OS/390 need to be compressed periodically. Use of PDSE structures reduces (or eliminates) this need. Also, PDSE data sets maintain the order of entries in their directories to improve performance compared to PDS structures.

- ***Librarian command language***

  The VSE Librarian consists of a single program, and its commands follow a consistent pattern. They are powerful and easy to use. They reflect the

functional enhancements of the VSE Librarian. The Librarian functions can be invoked through a console or through job streams (JCL).

In OS/390, each of the different utilities used to create and maintain PDS (or PDSE) data sets and their members has a different set of commands and somewhat different syntactical requirements. Combined with OS/390 JCL, the overall functions available are similar to the functions provided for VSE libraries.

- ***Interactive usage***

  The VSE Librarian runs in batch or interactive mode in static or dynamic batch partitions or interactive partitions of VSE/ICCF; it can attach VSE libraries/sublibraries dynamically through the VSE Librarian commands. This allows the user to move members from any VSE library/sublibraries into the ICCF library and vice versa. All Librarian services can be invoked from VSE consoles or ICCF terminals for all libraries.

  The interactive capabilities of the OS/390 Interactive System Productivity Facility (ISPF) are frequently used in OS/390 environments to perform many of the functions provided by VSE′s LIBR, such as creating and maintaining PDSs and their members. For example, it is common in VSE to edit a VTAM startup member (″B-Book″) in ICCF or CMS, and then submit a LIBR job to replace the system cataloged library member. In OS/390, ISPF facilities permit direct view, edit and replace functions for members in PDSs, so ISPF could be used to directly update the corresponding VTAMLIST member.

22.1.1 OS/390 ISPF Overview

ISPF also can be used in the following ways:

- As a general source code or document preparation and editing facility.
- To monitor and control program libraries.
- To communicate with MVS through TSO commands, CLISTs, or REXX EXECs.
- To develop a batch, interactive, or any other type of program and its documentation.
- To call dialogs that use Dialog Manager (DM) component and Program Development Facility (PDF) component dialog services to do the work of the application.

There are several functions in ISPF to view, browse, and edit partitioned data set members, as well as create and manage data sets. An even more powerful set of tools also exists in the Library Management Facility and Software Configuration Library Manager to manage multiple library levels.

Since ISPF usage is a key to productive use of your OS/390 system, we strongly recommend formal training in its features and the use of its functions. See the following ISPF books for orientation:

- SC28-1294 - *OS/390 ISPF Getting Started*
- SC28-1239 - *OS/390 ISPF User′s Guide*

22.1.2 OS/390 Library Management

The Software Configuration Library Manager (SCLM), a component of ISPF, introduces an object-oriented approach to library management and software development. The objects SCLM can control are members in partitioned data sets. SCLM keeps track of objects by means of multiple distinct VSAM files containing the accounting information of a member for a specific group.

The library management functions of SCLM are intertwined with SCLM's configuration management functions. The movements to and from the various partitioned data sets are examples of library management. The control over movements to and from the various partitioned data sets are examples of configuration management.

Chapter 23. LISTLOG/PRINTLOG - Printing Log Streams

Both VSE and OS/390 provide facilities to capture system log data in a hardcopy file, and means to display, print and archive it.

There are two utilities in VSE that help you print copies of your system hard-copy file and information about jobs that run in your system: PRINTLOG and LISTLOG.

In OS/390, the system hardcopy log can be saved on JES spool or in a log stream managed by the system logger, and printed by JES.

23.1 VSE PRINTLOG Utility

The IBM utility program PRINTLOG prints the hardcopy file from disk to SYSLST. Each line that appears on the screen of the display console is written to the hardcopy file, which resides on SYSREC. It may become necessary to print all or part of the hardcopy file. You may need to print the hardcopy log in order to review system events, or to determine which messages were issued for a certain partition. You should also print its contents before it is overwritten (see ″Hardcopy File Full Condition″ in the *VSE/ESA Guide for Solving Problems*).

23.2 VSE LISTLOG Utility Program

The LISTLOG utility program is used to gather information about how a particular job has run on the system. LISTLOG derives the information to be printed from the hardcopy file. It prints all messages and commands relevant to the partition in which the job ran. LISTLOG will provide a listing of the following items on SYSLST:

- job control statements which are written to the console
- console messages for the job
- operator responses for the job
- attention routine messages and commands issued while the job was running.

See *IBM VSE/ESA System Utilities* for additional information on LISTLOG.

23.3 OS/390 Hardcopy Processing

Hardcopy processing provides a permanent record of your OS/390 system activity and helps you audit the use of operator commands. You can record system messages and, optionally, commands, by using either the system log (SYSLOG), the operations log (OPERLOG), or an MCS printer. The group of messages and commands that are recorded is called the hardcopy message set. The system log, operations log, or MCS printer that receives messages is called the hardcopy medium. You can specify a group of console devices that can serve as backup devices for the hardcopy medium. You can also allow an extended MCS console to receive hardcopy messages from one or more systems in a sysplex.

The following describes the SYSLOG and OPERLOG and how to print them. See *OS/390 MVS Planning: Operations*, GC28-1760 for details.

23.3.1 SYSLOG

The system log (SYSLOG) is a data set residing in the primary job entry subsystem′s spool space. It can be used by application and system programmers to record communications about problem programs and system functions. The operator can use the LOG command to add an entry to the system log.

SYSLOG is queued for printing when the number of messages recorded reaches a threshold specified at system initialization. The operator can force the system log data set to be queued for printing before the threshold is reached by issuing the WRITELOG command. As part of the WRITELOG command, pick a SYSOUT class that is not used for normal printing so it is not printed unintentionally.

23.3.2 Printing SYSLOG

SYSLOG is so voluminous that you would not want to print the entire data set. Use SDSF to browse it and print portions by allocating SYSOUT data sets through JES.

Archive it using a spool archiving mechanism to save it for future problem determination and auditing purposes.

23.4 OPERLOG

The operations log (OPERLOG) is an MVS system logger application that records and merges messages about programs and system functions (the hardcopy message set) from each system in a sysplex that activates OPERLOG. Use OPERLOG rather than the system log (SYSLOG) as your hardcopy medium when you need a permanent log about operating conditions and maintenance for all systems in a sysplex. Only the systems in a sysplex that have specified and activated the operations log will have their records sent to OPERLOG.

The operations log is operationally independent of the system log. An installation can choose to run with either or both of the logs. If you choose to use the operations log as a replacement for SYSLOG, you can prevent the future use of SYSLOG; once the operations log is started with the SYSLOG not active, enter the WRITELOG CLOSE command.

In a single system environment, OPERLOG can reside on DASD, otherwise it is written to a coupling facility structure. When installation defined thresholds are reached, the system logger stores log data on DASD log data sets. You should use System Managed Storage (SMS) and DFHSM to manage the DASD log data sets. See *OS/390 MVS Setting Up a Sysplex*, GC28-1779 for details.

23.4.1 Printing OPERLOG

OPERLOG is so huge that you would not want to print the entire log stream. Use SDSF to browse it and print portions by allocating SYSOUT data sets for JES to print.

23.5 JES2 System Data Sets - Job Log and System Messages

The job's hardcopy log and any system messages related to the job are managed by JES2 as the "System Messages" for each job. Based on installation specifications and overridden on the job's JCL, they can also be saved or not based on the normal or abnormal termination of the job. These can be sent to a SYSOUT class that is held for viewing only and archived for permanent records.

23.6 Systems Management Recording

SMF formats the information that it gathers into system-related records and job-related records. System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information on the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session.

23.6.1 Printing SMF Records

IBM does not provide any utility to print SMF records, but there are plenty of ISV products that can be used to manage and format this information.

Chapter 24. VSE/Fast Copy and OS/390 DFSMSdss

The following briefly describes VSE/Fast Copy and the comparable OS/390 component, DFSMSdss.

24.1 VSE/Fast Copy (Online and Stand-Alone)

In VSE/ESA Version 1, VSE/Fast Copy runs stand-alone only. The on-line functions of VSE/Fast Copy have been incorporated into the program ″VSE/Fast Copy Data Set (VSE/Fast Copy).″ In VSE/ESA Version 2, VSE/Fast Copy (Online and Stand-Alone), was incorporated into the base code, VSE Central Functions. For information on VSE/Fast Copy see the *IBM VSE/ESA System Utilities*.

VSE/Fast Copy operates on volume-specific entities (IPL record, volume label, VTOC) and the set of files stored on the volume, taking the necessary information from the VTOC. A special Volume function is included for exceptional situations when the VTOC is no longer valid; this function processes an entire volume physically rather than being VTOC-driven (see ″Volume Functions″ in IBM VSE/ESA System Utilities).

With VSE/Fast Copy you can either move data directly from one disk to another or you may write it on intermediate tape to be restored later. The tape may either be unlabeled or have standard labels. When you restore the tape to disk you must also give the label option specified at the time of the tape creation. Alternate tape drives are supported.

VSE/Fast Copy stand-alone program can restore volume dumps, complete or partial, that were produced by the VSE/Fast Copy Data Set online program. To decrease the number of tapes, the VSE/Fast Copy dump and the stand-alone utilities may be on the same tape as the VSE/Fast Copy tape.

For disk volume identification, VSE/Fast Copy provides the following options:

- The volume identification of the input and/or output disk volume may be checked against the value specified in the utility control statement. This option checks whether you have mounted the correct disk pack. If not, you may proceed with the mounted disk pack, replace disk packs, or cancel the job.
- The volume identification written on an output disk volume may be:
  - copied from the original input disk
  - changed as specified in the utility control statement.

24.2 DFSMSdss - OS/390 Component

DFSMSdss has four functions to help you manage your DASD space:

- COMPRESS

  Compresses your partitioned data sets by taking unused space and consolidating it at the end of the data set. To make the unused space available for other data sets, you must use the RELEASE command. This does not apply to PDSEs.

- RELEASE

  Releases the unused space in sequential and partitioned data sets for use by other data sets.

- DEFRAG

  Consolidates the free space on a volume to help prevent out-of-space abends on new allocations.

- DUMP/RESTORE

  Deletes unwanted data sets and combines data set extents. (The COPY command can also be used to combine data set extents.)

Part 5. Setting Up the Migration Environment

Chapter 25. Prepare the Migration Environment

25.1 Introduction

Setting up the OS/390 environment, similar to setting up the VSE environment, involves installing the prerequisite hardware and software, and tailoring the system for your environment. As an example, we can describe it with the following steps:

1. Install and configure the required hardware.
2. Order and receive the OS/390 software along with all desired features, corequisite products, publications and so on.
3. Install the OS/390 software. For first time users of OS/390, we recommend that you install using SystemPac along with on-site services or other installation offerings. There are several services that provide more assistance such as SoftwareXcel Installation Express (SIE) in the United States.
4. Establish an inter-systems communication mechanism between your existing VSE system and the new OS/390 system for migrating and sharing data, programs and resources.
5. Set up your documentation, standards, operating procedures, training, and systems management mechanisms to manage and maintain the system.
6. Customize the OS/390 operating system, and necessary subsystems such as DFSMS, JES2, VTAM, RACF, and CICS. (Much of this may be done as part of the SIE installation.)

The following OS/390 documentation will help you get started:

- *OS/390 Introduction and Release Guide*, GC28-1725
- *OS/390 Planning for Installation*, GC28-1726
- *OS/390 MVS Hardware Configuration Definition (HCD) Planning*, GC28-1750
- *OS/390 Software Management Cookbook*, SG24-4775
- SystemPac, SIE, or ServerPac documentation.

In addition, here's a list of DFSMS manuals that can be used for implementing SMS:

- *DFSMS/MVS General Information*, GC26-4900
- *DFSMS/MVS Planning for Installation*, SC26-4919
- *Implementing System-Managed Storage*, SC26-3123

See *OS/390 Information Roadmap*, GC28-1727, and Appendix E, "Related Publications" on page 557 for a list of other documents.

OS/390 education for the systems programming staff is critical to the success of this installation. See IBM Education and Training's OS/390 course curricula for your area. Contact your IBM Representative or their Web site at the following URL:
http://www.training.ibm.com/ibmedu/roadmaps/mainframe/os390/.

25.2 Install and Configure Required Hardware

VSE and OS/390 operating systems both use the same basic S/390 hardware platform, although you will find that OS/390 may require more processor power, storage and DASD resources. On the other hand, OS/390 also provides more function, supports more devices, and is easier to manage as your applications and workload grow.

25.2.1 Processor Requirements

You will need a separate S/390 system such as a Multiprise 2000, or add another LPAR and supporting hardware to your existing processor. Customers with VM/ESA might want to add a virtual machine instead. If you add a migration and test load to your existing processor, you should add additional engines and memory to support the extra work.

Contact your IBM Representative to use one of the following capacity planning tools to size the processor requirements for your workload:

- LPAR/CE
- CP2000

Appendix B in *OS/390 Planning for Installation* describes the minimum processor requirements.

25.2.2 Devices Supported by OS/390

In general, all devices supported by VSE are supported by OS/390, except Fixed Block Architecture (FBA) DASD and most integrated communications adapters. See *FBA to ECKD Migration Aid - Internal Disk for the Multiprise 2000* which is a S/390 White Paper in the SG242000 PACKAGE on MKTTOOLS. Contact your local IBM representative for a copy.

See Appendix B in *OS/390 MVS Hardware Configuration Definition (HCD) Planning*, GC28-1750 for a complete list of supported devices.

25.2.3 DASD Requirements

Direct Access Storage Devices (DASD) are required for your OS/390 System Libraries, Page, Spool, Programs, Data, and Work/Public/Storage volumes. If you are configuring your own system, see Chapter 4 and Appendix E in *OS/390 Planning for Installation* for a thorough description and recommended pack layout. If you are using SystemPac or SIE to build your system, the volumes will be configured slightly differently for you.

While it is technically possible to create a one or two-pack OS/390 system, you will need a lot more DASD for productive use. The initial number of volumes required for system data sets with a SystemPac may be two RES, two DLIB, one SMP, and one CAT volume. Even though these contain paging and spool data sets, you will very quickly run out of space if you try to do any work.

Most installations require more spool space on OS/390 than they did with POWER because of the allocation units (track-groups) and space reclamation differences. (You will want to automate some method for purging old spool data sets.)

A beginning rule of thumb shows 12 volumes of 3390-3 (or equivalent) DASD, allocated as follows: **(Your mileage will vary!)**

Table 45. OS/390 DASD Layout

Volume Use	**Number**
System Libraries (RES)	2
Distribution Libraries (DLIB)	2
SMP/E Work (SMP)	1
Catalogs (CAT)	1
Paging Data Sets (PAG)	1
Spool & Checkpoint (SPL)	1
Softcopy Library (BKM)	1
DFSMShsm ML1	1
Storage/Work/DFSMS Volumes	1 or more
User Program/Data Libraries	1 or more
ISV Products	0 or more
TOTALS	**12 or more**

These numbers are very dependent on the installation, and will increase dramatically at the end of a ″mass migration″ in order to duplicate user data files.

25.2.4 Other Hardware Requirements

For the most minimal testing, you will need at least the following devices, depending on the number of users, and the size of your applications and databases.

Tapes Tape drives will be required for system dumps, backups and restores, application testing, and DFSMShsm ML2 migrations. These can be switched between the production VSE system and the OS/390 system, but you should plan on at least two dedicated tape drives for the OS/390 system.

Console You should have at least one console connected through non-SNA control units for system operation and a second console for backup and operator training.

Printers You will occasionally need to print. Printers can either be dedicated to the OS/390 system, switchable from the VSE system, shared on a LAN, or accessed on the VSE system via NJE. RJE printers are also an option if you already plan to have remote workstation printers.

Remote Workstations
If you are migrating remote workstations to JES2 RJE, it may be very helpful to have additional workstations dedicated for testing.

Communication Controllers
You need to provide for remote access of TSO, RJE, NJE and application (for example, CICS) users. With multiple channel adapters, you can also allow terminals connected to your VSE system to access

OS/390 through cross-domain resource definitions. They are also useful for data transfer via NetView FTP or NJE. OSA (Open System Adapters) are often the most economical solution.

You will want access to other devices in your installation. They can be switchable or connected via a common local area network (LAN). See the *OS/390 MVS Recovery and Reconfiguration Guide*, GC28-1777 for more configuration planning information.

25.2.5 Inter-Systems Connectivity

You will need to share files and I/O devices between your VSE system and the new OS/390 system. Users on the new OS/390 system need access to data and resources such as printers and interactive terminals on your existing VSE system, and VSE users need to send programs and data over to the new system for migration and testing.

25.2.5.1 Shared DASD

It is both difficult and dangerous to share DASD between VSE and OS/390 systems. Difficult because they don't support the same file organizations. Dangerous because there is no serialization mechanism to prevent multiple updates or data corruption from occurring.

However, under strict manual controls (for example, vary online/offline) you can set up some common DASD for sharing data and programs between your VSE and OS/390 systems. This way, you can avoid an intermediate transfer of the data to tape or sending it via communication mechanisms such as NetView FTP.

Since VSE doesn't support indexed VTOCs, a volume with an indexed VTOC must be converted to a non-indexed VTOC (OS VTOC) before transporting it to the VSE system. Chapter 5, "Disk and Tape Storage Considerations" on page 97 chapter has more information about DASD sharing.

25.2.5.2 Tape Drives

You will need some tape drives to transfer large amounts of data between the two systems. They can be switched between the two systems, although you probably will want to dedicate at least two tape drives to the OS/390 system.

25.2.5.3 Terminal Access

You will need to provide terminal access for TSO users on your new system. This can be done in several different ways:

- Dedicate terminal controller to the OS/390 system.
- SNA cross-domain logon from your existing terminals on your VSE system using VTAM-controlled CTCs
- SNA cross-domain logon through a Communications Controllers (for example, 3745) shared with your VSE system using multiple channel adapters or EMIF.
- Use a Token-Ring network shared with the VSE system, and an OSA (Open System Adapter) on the new processor.

See 25.5.1.3, "Providing Terminal Access to the OS/390 System" on page 414.

25.2.5.4 Data Transfer and NJE

You will want to set up an NJE connection between the two systems for remote job submission and for routing print files or bulk data between the two systems. Use the same communication controllers, real Channel to Channel adapter (CTC) controlled by VTAM or virtual CTCs. You have several choices with ESCON CTCs, virtual CTCs under VM, and 3088s.

See 25.5.1.5, “Providing NJE Connection to the OS/390 System” on page 415.

We recommend NJE for job submission, spool file and printer transfer, and disk/tape for file transfer and VTAM cross domain for terminal access.

25.3 Order and Install the OS/390 Software

As with VSE, there are several options available for you to order your OS/390 system and install it. Some assume that you already have a running OS/390 system, while others provide a ″starter system.″ Some are ″entitled″ or included with the price of the OS/390 software, while others are ″fee-based″, and include on-site IBM assistance, and integrate ISV (Independent Software Vendor) products on your system.

In general, we recommend the SoftwareXcel Installation Express (SIE) for the US customers migrating to OS/390, and full volume dump format SystemPac for the non-US customers.

OS/390 Planning for Installation, GC28-1726 is your primary reference book for this. Although it is oriented towards ServerPac, Chapter F ″Checklist of Installation Procedures″ provides an excellent list of planning activities. Each of the following installation options includes its own planning materials. You can obtain on-site assistance with some of the offerings, or separately.

Note: Not all options are available in all countries! Refer to the most recent IBM Announcement Letters or your IBM representative for details.

25.3.1 Fee-based Methods of Installing OS/390

For the first-time OS/390 user, you should consider one of the following fee-based IBM services.

25.3.1.1 SoftwareXcel Installation Express (SIE)

This is an IBM US offering which provides pre-built OS/390 system packages in full volume dump format, tailored to customer hardware and software configurations. SIE includes on-site planning, installation, and package testing by an experienced IBM Technical Representative. You can also obtain a compatibility research report and selected non-IBM software products integrated into the system package.

Contact your local IBM Representative for more information.

25.3.1.2 SoftwareXcel SystemPac/MVS

The SystemPac installation offering is a world-wide offering similar to SIE, but without on-site assistance by IBM. You can use this to tailor OS/390 to your environment (such as DASD layout, migration of MVSCP/IOCP to IODF, and naming conventions) based on information provided to IBM. With this offering, selected non-IBM products can be integrated. This offering can be delivered in IEBCOPY dump-by-data-set format or in full volume dump format.

Use this in conjunction with IBM services to set up and tailor your OS/390 system. See *Custom-Built Offerings Planning*, SC23-0352 and *CustomPac Installation Dialogs*, SA22-7240 for more information.

25.3.1.3 Other Offerings

There are many other fee-based offerings based on SystemPac available in specific countries or geographies. Many are packaged with on-site assistance to help install and tailor the system, provide services, maintenance, and financing to help customers get to current technology.

Other fee-based help includes Washington System Center services, customized solutions, hardware services, and software services. Contact your IBM Representative for details about additional installation services.

25.3.2 Entitled Methods of Installing OS/390

These are only recommended for installations which have a running OS/390 system and want to make incremental upgrades to it. There is no FunctionPac, ProductPac, Custom Built Installation Process Offering (CBIPO), or stand-alone product tape for OS/390.

If your OS/390 was not tailored to your hardware configuration, you will need to use the Hardware Configuration Definition (HCD) to define I/O configurations to both the software and hardware. HCD can be used as part of the initial ordering procedure for many of the above offerings to create an input/output definition file (IODF). If starting with your VSE configuration, you can add the necessary MVS control statements to your IOCP and convert them to HCD.

Both of the following methods require OS/390 or MVS as the driving system.

25.3.2.1 ServerPac

This software delivery package consists of installed products and integrated service for a ready-to-IPL system. To install, you use the CustomPac Installation Dialog -- the same dialog that is used for all the CustomPac offerings, including SystemPac and ProductPac. You can use ServerPac to install a new OS/390 system, replace an entire existing system, or replace an existing system except for its operational data sets (SYS1.PARMLIB, SYS1.PROCLIB, and the like).

You can order other IBM products and subsystems in a single ServerPac order. However, CICS, IMS, DB2, and NCP products will be delivered in separate ServerPacs.

See *ServerPac: Using the Installation Dialog*, SC28-1244.

25.3.2.2 CBPDO

Custom-Built Product Delivery Option is a software delivery package consisting of uninstalled products without integrated service. You must use SMP/E to install the individual OS/390 elements and features, and their service, before you can IPL.

This method is not recommended for the new OS/390 installation.

25.4 Set Up Standards, Procedures, and Documentation

You now have a running system that is tailored to your environment and users. Your next step is to set up and document standards and procedures for all those people that will be using and operating it.

25.4.1 Installation Standards

As you develop your new OS/390 environment and your applications and user community grow, it is very important to develop good standards for all your resources and users. It is much easier to do this from the beginning than to go in later and impose standards and procedures were there were little or none beforehand.

25.4.1.1 Data Management Standards

DFSMS naming standards are not trivial, but there is a lot of guidance. This is really part of the DFSMS implementation process, which is a whole study in itself. The DFSMS FIT Redbooks have suggestions for naming the constructs and worksheets to assist with the migration. A good rule, as always, is to keep it fairly simple. Here are some suggestions for name-significant characters:

- Data Classes - Start the name with D or DC, and include DSORG, RECFM, LRECL, or Space requirements.
- Storage Classes - Start it with an S or SC. Examples are SCSTAND, SCPREF, SCFAST, and SCNOSMS. Distinguish service but don′t use parameter values.
- Management Classes - Start the name with M or MC. Use the remaining characters for indicating which service elements separate it from the other classes. For example, MCNOMIG for data sets that you don′t want to have migrate/recalled. Other attributes could include Backup, Archive, Migration, and Space attributes.
- Storage Groups - Start the name with G or SG. The name should identify the type of data associated with the pool. For example, use things such as SGWORK (for temp), SGPRIME (for batch production), or define storage groups according to size, for example SGLARGE for large and SGSMALL for small data sets. The advantage to this is reducing fragmentation on the volumes and reducing out-space abends for new allocations and extents.

Storage systems education is available for your systems programming staff. See IBM Education and Training′s storage systems course curricula for your area. Contact your IBM Representative or their web site at:
http://www.training.ibm.com/ibmedu/roadmaps/mainframe/storsys/

Related Redbooks

Here is a list of DFSMS ″Fast Implementation Techniques″ (FIT) Redbooks:

- *DFSMS FIT: Fast Implementation Techniques Process Guide*, SG24-4478
- *Get DFSMS FIT: Fast Implementation Techniques*, SG24-2568
- *DFSMS FIT Forms and Foils*, SG24-2570
- *DFSMS FIT: Fast Implementation Techniques Installation Examples*, SG24-2569
- *DFSMS/MVS V1R4 Technical Guide*, SG24-4892

25.4.1.2 MVS Naming Standards

The following OS/390 resources are all identified by names. Some names are seven or eight characters, others can be up to 16 or 44 characters in length. By using significant positions of the name, you can more easily manage and control them for registration, security, and general systems management purposes.

Most installations use the first character of the name to identify the resource type or production application, such as **P** (Production), **T** (Test), **S** (Systems Programming), and **I** (Inventory Applications).

There are lots of entities to name in MVS. It is a good idea (if not required) to start these names with an alphabetic character (A-Z). This is not a complete list, but below are some resources that need names.

Data Sets

Names can be up to 44 characters long and start with a high-level qualifier that identifies both who owns the data set (such as a user ID, project, application, or group), along with an indication of production, test, or systems. Use these long names, with specific levels of the DSName to indicate the following:

- System vs. Production vs. Test; Temporary vs. Permanent
- Application Name or ID
- Version Level

The data set name doesn′t need to identify the access method being used. (The main reason people used to do this was because of the old VSAM catalog volume ′ownership′. Since this is no longer an issue with ICF catalogs, there is no need to include this.) Some other things ***not*** to include in the data set name because they will probably change: department, location management criteria, device type, or expiration date.

Generation Data Groups

The only difference here is that the data set name loses one level of qualification, the lowest level. Don′t use the generation to indicate a different type of data. For example, don′t use ′(+1)′ for reports and ′(+2)′ for intermediate files. Have separately named GDGs instead.

DASD and Tape Volume Serials

DASD and tape volumes are typically labeled so that they can be logically related to an application, geometry, storage group, or purpose, for example TSOxxx or PAYxxx. Keep in mind how you are going to list volumes from ISMF. For example, if I want to list all of my work or TSO packs, it would be nice to simply enter WRK* or TSO*.

Other MVS Names

There are many other objects which require naming in OS/390. Here is just a sample:

- Users (TSO, online, batch, operators)
- Job names
- Job step names
- JCL procedure names and proclibs
- Application program names
- WLM service classes
- WLM resource environments

25.4.1.3 JCL Standards

You should set up JCL (job control language) standards for your batch jobs, as well as started tasks and TSO logon procedures. Here is a partial list of some of the attributes to include in your JCL rules:

- Job Attributes
  - Job names
  - Accounting information
  - Job classes
  - MSGCLASS and MSGLEVEL
  - Specification of USER, GROUP, SECLABEL, and PASSWORD
  - Use of JOBCATs, JOBLIBs
  - Estimated lines, bytes, pages, and cards
  - Use of PRTY, PERFORM, REGION, TIME parameters
  - Job restart options
  - Use of commands in the jobstream
- Step Attributes
  - Job step names
  - Use of ACCT, DPRTY, PERFORM, REGION, TIME parameters
  - Use of STEPLIBs and STEPCATs
- Data Set Attributes
  - Unit names
  - GDGs
  - Tape standard labels
  - DASD space units and parameters
  - Output classes and printer attributes
  - Output routing destinations

25.4.2 Systems Management Procedures

In general, you can use many of the same processes that you are currently using. There are tools and products specific to OS/390 such as SystemView, TME 10, and INFO/MAN. Many products that work with VSE also work on OS/390.

This section addresses some of the basic elements associated with good management practices of running a system. See Chapter 30, “Systems Management Philosophy and Methodology” on page 457 for a more complete discussion.

25.4.2.1 Enforcing Installation Standards

You will continue to refine the standards developed above, and should use RACF to protect critical resources. You can also use installation exits to enforce standards not controlled by RACF, but keep in mind that it is often easier to enforce them through other procedures.

25.4.2.2 Implementing System Security

OS/390 users have access to all data sets in the system unless specifically restricted via the security product. Intentional or unintentional modification of system data sets can compromise system availability.

You should use RACF (now called the OS/390 Security Server) to protect critical resources such as system data sets, catalogs, and access to valuable, sensitive or confidential data. You should identify all users of the system, whether they are TSO, on-line, batch job owners, or console operators.

Security (RACF) can also be used to enforce the installation standards. See Appendix D “Security for System Data Sets” in *OS/390 Security Server (RACF) Security Administrator*, SC28-1915.

See the following RACF books for more information:

- *OS/390 Security Server (RACF) Introduction*, GC28-1912
- *OS/390 Security Server (RACF) Planning: Installation and Migration*, GC28-1920
- *OS/390 Security Server (RACF) General User's Guide*, SC28-1917
- *MVS 3.1.3 and RACF 1.9 Security Implementation Guide*, GG24-3585
- *RACF V2.2 Installation and Implementation Guide*, SG24-4580

25.4.2.3 Backing Up Your System

Periodic system backups are critical to maintaining access to critical resources and protecting your investments in systems programming and migration efforts. You should take full-volume dumps of all system packs except paging data sets, and JES2 spool and checkpoint volumes. This should be part of your SMS strategy, and developed early in the project along with your disaster recovery goals and requirements.

25.4.2.4 Creating an Emergency Backup System

As careful as you are, there may be a time that you can not IPL your OS/390 system because the IPL text in the SYSRES is damaged, the master catalog is deleted, or the JES2 procedure has a JCL error. (There are many other reasons why you may not be able to IPL.)

In any case, you should have a backup OS/390 system that you can IPL to diagnose and fix the problem. Many installations have a simple one or two-pack MVS system that can support a TSO user and a few batch jobs. Another strategy is to use your system maintenance environment to provide a simple but usable backup system. There are also vendor products such as SAE, which provide ″rescue″ systems.

25.4.2.5 Setting Up Critical Operations Procedures

You should set up, test, and document procedures that are critical to the smooth operation of your system. Here is a sampling:

- System IPL, JES2 warm-start
- JES2 checkpoint reconfiguration
- JES2 ABEND and hot-start
- System shutdown
- VTAM startup and shutdown
- VTAM vary node active and inactive
- Switching devices between systems
- System recovery, restart, and first-failure data capture
- Stand-alone dumps
- Managing spool space and spool full conditions
- System backup (full and partial)
- Emergency system restore

See Chapter 28, “Orientation to OS/390 Console Operation” on page 443 for some examples of OS/390 console operation.

25.4.2.6 Managing Change

As you apply maintenance and make changes to your configuration, follow these simple rules:

- Make changes incrementally, not many at once.
- Test the effects of each change to make sure you do not regress your system.
- Make sure you can back out changes in the event of a problem.
- Document all changes, don′t rely on your memory.
- Adhere to the age-old ″Keep it Simple (KISS)″ philosophy to minimize unnecessary complexity in your system.

These guidelines are critical to the migration project, especially as you approach the switch-over time.

25.4.2.7 Managing Problems

Staying on top of problems is important, especially with a project as massive as converting from VSE to OS/390. Use a process and tools with which you are familiar, perhaps what you are currently using with your VSE system. In addition, you might want to set up a “strategy room” and large marker boards for managing problems at switch-over time.

In addition, we recommend you keep your software service at a current level to minimize the possibility of rediscovering old problems. See 25.5.1.2, “Applying Preventive Service” on page 414.

25.4.3 Documentation

You should already have the following planning publications which are available as part of the *OS/390 Installation Planning Kit*, GK2T-6710:

- *OS/390 Planning for Installation*, GC28-1726
- *OS/390 Introduction and Release Guide*, GC28-1725
- *OS/390 Information Roadmap*, GC28-1727

The Information Roadmap will then list the other publications you may need.

25.4.3.1 Your Hardcopy Library

Some books you will need to keep as hardcopy versions available, such as:

- Planning Books listed frequently in this bulletin.
- Other OS/390 books you will need to review frequently, or take home to read.
- MVS and JES2 Operator Command books should be kept at the system console, as well as all the Messages books.

25.4.3.2 Your Softcopy Library

There are so many books in the OS/390 library that you will want to get familiar with the softcopy library and the BookManager/Read tools for viewing information. This is extremely useful for searching for and finding the right book.

The *OS/390 Online Collection*, SK2T-6700 is available on CD-ROM and updated quarterly. (It is also available on tape as a feature of OS/390.)

The CD-ROM can be used on a PC workstation with OS/2 or Windows, and uploaded to DASD and used with BookManager READ/MVS. In general, we recommend the MVS platform for normal softcopy viewing, but you should have a set of OS/390 CD-ROMs available for viewing on a PC workstation in the case of an emergency or system outage.

Printing Softcopy Books

Use the Softcopy Print facility in OS/390 to print softcopy BOOKs from BookManager READ/MVS. With the BookMaster GML option you get the nicest looking printout, and it is the easiest to implement. OS/390 has included enough code from DCF, PSF, BookMaster, and the required AFP fonts.

See *OS/390 V2R4.0 Printing Softcopy BOOKs*, S544-5354 for more information.

Redbooks

You should have the *S/390 Redbooks Collection*, SK2T-2177 which has over 300 technical bulletins in BookManager format related to S/390. They are written by the ITSO and Advanced Technical Support Systems Centers.

25.5 Customize Your New OS/390 System

Before you start using your new OS/390 system, you must complete the installation and tailoring process by customizing the system for your use. Depending on what method you used to install the software, some of the items listed below may already have been completed for you or you may have contracted for some additional service assistance to perform these items. The following items are not a complete list, but can be used as a general checklist:

__ Install remaining IBM Service

__ Exercise IVPs

__ Back up system, and test recovery by restoring it

__ Update VTAM and NCP definitions, generate the NCPs and test them

__ Update MVS SYS1.PARMLIB definitions and test them

__ Tailor SYS1.PROCLIB procedures and test them

__ Update JES2 parameters and test them

__ Update DFSMS parameters and test them

__ Update RACF security environment and test it

__ Update LE, language options and procedures and test them

__ Define TSO/E user IDs and test them

__ Define user catalogs and test them

__ Set up SMP/E maintenance environment and test it

__ Set up IPCS service aids and test them

__ Set up CICS and other application subsystems and test them

__ Set up application development environment and test it

__ Develop operations procedures for common and critical tasks

__ Test operations procedures and train operators

__ Set up installation standards, test and document them

__ Set up hardcopy and softcopy documentation libraries

__ Set up accounting and billing procedures

__ Once more, exercise IVPs, and back up system

If you need additional assistance to complete your production ready system, you may want to contract for some services work to perform these items. The fees for these items would be based on exactly what work was required. You should ensure that all items performed as part of the services are documented so that you can make any needed adjustments in the future.

25.5.1.1 Verifying the New OS/390 System

Before you begin tailoring the new system, and at each stage of the tailoring process, you should verify that the system is in good shape before you take the next step. If your system was installed as part of a packaged offering, it was probably verified before you received it.

IBM′s comprehensive testing does not replace the need for this testing in your own environment. Here are some sample steps copied from the OS/390 Checklist:

__ ***1.*** Initialize the system (IPL)

__ ***2.*** Initialize JES2

__ ***3.*** Log on to TSO/E

__ ***4.*** Run the installation verification programs (IVPs)

__ ***5.*** Submit a job and check its output

__ ***6.*** Sign on to a terminal with CICS or IMS and initialize a region

__ ***7.*** Submit a CICS or IMS transaction and look at the results

__ ***8.*** Run a complex application with known data and measure its elapsed time and resources

__ ***9.*** Check for completeness of accounting records

__ ***10.*** Test non-IBM product functions

IVP jobs are listed in your CustomPac documentation, or in *OS/390 Planning for Installation*. This list is not complete and should be tailored for each installation based on the importance of each function, likelihood of errors, and expanded as experience dictates.

25.5.1.2 Applying Preventive Service

You should update your OS/390 service level to a fairly current level and run through one more verification test before you switch your production over to it. This is especially true if the migration project was long, and you have not been applying maintenance on a regular schedule.

MVS Recommended Service Upgrade (MVS/RSU) is a preventive service philosophy for all OS/390 and MVS products. MVS/RSU reduces the volume of PTFs you must apply for preventive maintenance and reduces the chance of encountering a PTF in error (PE), resulting in a more stable system.

IBM recommends that you APPLY all MVS/RSU PTFs on your OS/390 system. However, the customer must make the final decision as to what service will be installed.

CustomPac offerings (that is, SystemPac, FunctionPac, ProductPac, and ServicePac) will continue to follow the current CustomPac Service philosophy based on PUT levels combined with RSU levels.

See the *OS/390 Software Management Cookbook*, SG24-4775.

25.5.1.3 Providing Terminal Access to the OS/390 System

You can connect the VTAM subareas in the VSE and OS/390 systems together and use cross domain resource sharing to access OS/390 applications from terminals connected to your VSE system (or vice versa). See Chapter 16 ″Implementing a Subarea Network″ in *VTAM Network Implementation Guide*, GC31-8370 for details.

See also the samples provided in *IBM Network Products Implementation Guide*, GG24-3649.

25.5.1.4 NetView FTP Access

You can also use the same VTAM connections to send bulk data between the two systems with NetView FTP.

See *NetView FTP V2 MVS Installation, Operation, and Administration*, SH12-5657.

25.5.1.5 Providing NJE Connection to the OS/390 System

You can connect VSE/POWER and OS/390 JES2 systems together via NJE to route jobs and output from one system to the other. In addition to the POWER and JES2 books, see *NJE Installation, Operation and Use with JES2 and Other Systems*, GG22-9339 for guidance and examples.

25.5.2 MVS BCP Customization

There are many parameters and installation exit points which you can use to further customize your OS/390 system. The following information is not a complete list, but a short overview of the members in Parmlib and exit points which may be used.

See the *OS/390 MVS Initialization and Tuning Guide*, GC28-1751 for storage considerations, paging, and SRM guidance.

See Chapter 1 of *OS/390 MVS Initialization and Tuning Reference*, GC28-1752 for general information called ″System Tailoring″:

- MVS Hardware Configuration Definition
- System Tailoring at Initialization Time
- Understanding the Master Scheduler Job Control Language
- Overview of Parmlib Members
- Implicit System Parameters
- Managing System Security -- APF-Authorized Library List
- Specifying Installation Exits
- Specifying LNKLST Concatenations

25.5.2.1 SYS1.PARMLIB Parameters

There are dozens of ″named″ members of SYS1.PARMLIB for OS/390 Release 4 customization. They are described in the *OS/390 MVS Initialization and Tuning Reference*.

There are also many other members that can be used to tailor base elements of OS/390, optional features, other IBM products, and ISV products. See the appropriate product documentation for specifics.

25.5.2.2 MVS Exits

There are many installation defined exit points in OS/390 components where installations can insert their own code to affect normal processing. There are exits for SMF, DFSMS, IPCS, JES2, RACF, TSO/E, MPF, and many other functions. However, these are not usually necessary and can be more trouble than they are worth. (Exits often need to be re-assembled or re-worked with future system upgrades.)

25.5.2.3 Tailoring Other Components

Other features of OS/390, such as JES2 are described in other chapters of this book. The elements of OS/390 are listed below and each has its own set of books on installation and customization.

25.5.3 Other OS/390 Elements

OS/390 is made up of base elements and optional features. Here is a list of the pieces you may have with OS/390 Version 2 Release 4. Check the latest OS/390 announcement letter, or one of the following books for a complete list:

- GC28-1725, *OS/390 Introduction and Release Guide*
- GC28-1726, *OS/390 Planning for Installation*
- GC28-1963, *OS/390 Parallel Sysplex Test Report*

There is also a list in 2.2, “OS/390 Components/Products/Subsystems” on page 18.

25.5.3.1 Base Elements for Release 4

These components all come with your OS/390 system and are enabled, ready to use:

- System Services - DFSMSdfp, JES2, ISPF, ICKDSF, HLASM, TSO/E and so on
- Systems Management Services - HCD, ICSF, SMP/E, and SystemView base
- Application Enablement Services - DCE AS, Encina Toolkit Executive, GDDM/MVS (PCLK & OS/2 Link), LE, OS/390 AET, SOMobjects RTL, & VisualLift RTL
- UNIX Services (X/Open UNIX 95 functions)
- Distributed Computing Services - DCE base, DCE DFS, and DFSMS/MVS NFS
- Communications Server - FFST, IBM TCP/IP, TIOC, and VTAM
- LAN Services - LANRES, LAN Server, and OSA Support Facility
- Network Computing Services - BookServer, and Lotus Go Webserver
- Softcopy Services - BookManager READ/MVS, and Softcopy Print

25.5.3.2 Optional Features for Release 4

Also included in your OS/390 system are the following optionally priced features. You can use them by registering them according to the *OS/390 Program Licensed Specifications*, GC28-1728. See the OS/390 Announcement letter, and *OS/390 Planning for Installation*, GC28-1726 for details.

- System Services - JES3, MVS/BDT NJE & File-to-File
- Security Server - RACF, and DCE Security Server
- Systems Management - DFSMSdss, DFSMSrmm, DFSMShsm, HCM, RMF, and SDSF
- Application Enablement Services - DFSORT, GDDM-PGF & REXX/MVS, IBM C/C++ Compiler, IBM HLASM Toolkit, LE DES, SOMobjects, and VisualLift
- Distributed Computing Services - DCE User Data Privacy, and IP PrintWay/NetSpool
- Communications Server - IBM TCP/IP Kerberos, NPF, and OS/2 Offload
- Network Computing Services - Lotus Go Webserver
- Softcopy Services - BookManager BUILD/MVS

25.5.3.3 Independent Software Vendor Products

If you chose to use ISV (non-IBM) products, you should recognize the additional implementation, customization, maintenance and tuning requirements that accompany them.

Chapter 26. Test Environments

This chapter describes the different needs for test systems during and after the migration to OS/390. There are many valid test configurations which vary according to your installation's testing and maintenance philosophies, as well as your environment.

Here are some of the discussion points:

- VM, LPAR, or stand-alone test systems
- Number and availability of test systems
- Inter-connectivity and isolation of test systems
- Shared DASD or cloned data between production and test systems

References for additional information: See the *OS/390 Software Management Cookbook*, SG24-4775 for some more detailed advice on how to manage software levels.

26.1 Introduction

Just as you need a test VSE system to test out maintenance and new releases, you will need a test OS/390 system when you go to OS/390. As part of this conversion, you will also need test OS/390 systems for systems programmer tailoring and testing (a "sand-box"), application conversion and testing, and fallback or rescue situations.

26.1.1 Differences in Testing "Philosophy"

As your production system's availability requirements get more demanding, so will your need for a separate test system.

You also will want to apply maintenance and upgrade your system more frequently with OS/390 than you did with VSE.

26.1.2 Terminology

The term "Test System" means different things to different people. Any system that is not a "Production System" required for supporting the primary business of the company is a "Test System."

There are different test system requirements and many ways to set them up. Here is a sample set of three different test OS/390 systems during the life of the conversion project. They will also be the basis after you go into production on OS/390.

Backup System A system that is usually not running, but available to IPL should one of your systems become inoperable. (Also called the **Rescue** or **Fall-Back** system.)

This system should have a minimum number of volumes, and can be saved to tape and restored by stand-alone DSS.

Application Development & Test System

A system that is expected to stay up without disruption at least during ″normal working hours″. Understand that any outages to this system will affect your applications conversion and testing activities. This logical image may eventually become your production OS/390 system when you cut-over.

Maintenance System The next version of operating system software, or next round of maintenance that will be rolled forward into production. (It is very important to use SMP/E to maintain your OS/390 system.)

This can double as your “Systems Programmer Sandbox” for testing new operating functions or options that can be brought up and down as desired without worrying about disrupting other users.

This can also be your system on which operators can be trained without fear of reprisal should they do something wrong. This system can be re-IPLed during scheduled times to train operators on startup and shutdown procedures.

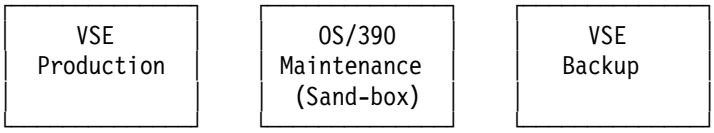
After your production cut-over to OS/390, you will also need a “Fall-Back System” for backup should the primary production system have too many problems to keep it up. This is different from a “Backup system” mentioned above, and may just be a copy of the SYSRES volumes and program libraries at the previous level of maintenance.

26.2 Test Systems in the Life of the Migration

You will need at least three S/390 images during this migration. Here is a simple example of how they can be used at different stages of the migration:

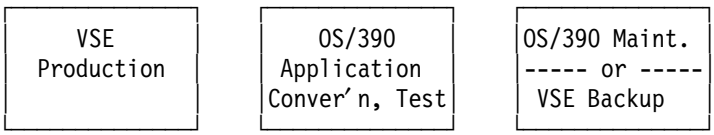
1 **Initial OS/390 System Installation, Tailoring, and Verification**

During this phase, the test system is undergoing frequent changes and may have to be restarted often.



2 **Application Program, JCL, and Data Conversion**

During this phase, the test system must be readily available for the programmers and others to make and test changes to the applications. You will also need a systems programming system for applying maintenance, and testing new or modified systems.

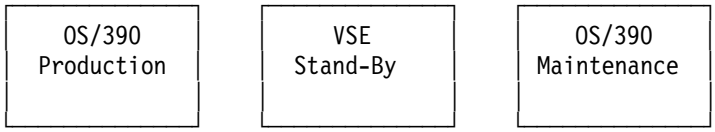


3 **Final System Test on OS/390**

Just before you migrate to OS/390, you should run all your important applications in parallel, using the same environment as above. Compare the results of both systems to make certain there are as few surprises as possible.

4 **Final Production Cut-Over to OS/390 (″D″ Day?)**

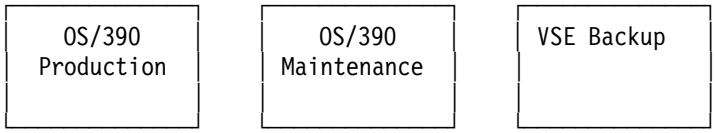
When you finally migrate your production applications to OS/390, you will need a backup VSE system standing by for emergency rerun of applications that uncover any conversion problems after you go live.



5 **After Production Cut-Over to OS/390**

Once you are in production, you still need an on-going test system environment for applying maintenance, and testing new releases of OS/390 and subsystems.

Even after the migration is complete, you will still want to keep a backup VSE system around for emergency, but this requirement will fade over time.



26.3 VM, LPAR, or Standalone Systems

Now that we have sketched briefly the number and types of operating system images that will be involved in this migration, we need to consider a very important question. What is the best way to implement these multiple system images for the migration period, and perhaps into the future given the need for test and backup OS/390 systems? When considering implementation of multiple system images the following set of choices exist:

- Separate hardware platform for each system image (included here would be consideration of using P/390s to support single system images)
- Physical partitioning of a single or multiple hardware platforms
- Logical partitioning of a single or multiple hardware platforms
- Software partitioning of a single or multiple hardware platforms
- Some combination of the above choices

The choice you make from the above set depends on many variables such as: your current hardware environment, the hardware environment you may be migrating to, your current and future software environment, the physical space you have available for hardware, your hardware and software budget, the skill set of your I/T staff, and so on. Each of the choices listed above has positive and negative aspects depending upon how your environment maps to the variables described. It would be possible to enter into a lengthy discussion of how to

evaluate each of the alternatives mentioned, but that would be beyond the scope of this book, and not necessarily relevant to the task at hand.

Since this document is directed toward migration of VSE systems to OS/390 primarily on CMOS processors, we will not consider the choices of physically partitioning one or more processors, since this option is not available on CMOS processors. In addition, we will not consider the choice of using separate hardware platforms for each system image since it is very often not a very practical alternative. Although, with the P/390 platform it is a much more attractive possibility than it ever used to be. Rather, we will limit our consideration to the choices of logical partitioning, software partitioning, or a combination of both.

An excellent comparison of the relative merits of logical partitioning and software partitioning with VM/ESA is contained in the IBM Redbook *ES/9000 Multi-Image Processing from a VM/ESA Perspective Volume 1*, GG24-3920. The reader is directed to this reference for a comprehensive comparison of LPAR and VM/ESA. The comparison provided is neutral in the sense that it is not considering any particular task other than running and supporting guest operating systems. In other words, the added consideration of which solution best supports a migration environment is not included. Thus we will attempt to add to that discussion from the perspective of migrating VSE systems to OS/390. It should be noted that this Redbook was written in late 1993, and as such is out of date in terms of hardware platforms discussed. However, the bulk of comparison items, and conclusions drawn are still very relevant to today′s hardware platforms. Details such as the number of partitions supported on a particular hardware platform have changed, however, the basic notion that a finite number is supported through logical partitioning, and a substantially larger number of images is supported through software partitioning still holds true.

Before presenting a recommendation for the migration environment, it would be good to review briefly the characteristics of logical and software partitioning. This review will serve as a basis for the recommendations that follow.

26.3.1 Logical Partitioning

Logical partitioning is achieved through the use of Processor Resource System Manager (PR/SM) licensed internal code (LIC). Processors since the 3090 series have had this LIC available. However, in the 3090 days, it was most commonly a priced feature added on to the processor. With the general availability of the ES/9000 series processors, PR/SM has become a standard feature on S/390 processors. Multiple system images are supported by defining logical partitions, and assigning resources (CPU, Central Storage, Peripherals) to the partitions. This is accomplished through specifications made in the IOCP, and commands entered through the hardware system management console.

It is possible to define more partitions than the available physical resources can support, however, the number of partitions that can be active at any time is limited to what the physical resources can support. For example, if your Central Electronic Complex (CEC) has 1GB of central storage available, and you have defined five logical partitions each specifying 500MB of central storage, you are limited to activating two of the partitions at any one time. The remaining three partitions are defined, but cannot be activated until one of the currently activated partitions is deactivated.

Recent advances in the PR/SM LIC have solved some of the real storage management difficulties encountered in the past concerning assigning contiguous storage chunks to LPARs. In addition, ESCON Multi Image Facility (EMIF) channels have reduced the number of physical channels that must be installed to support multiple partitions by allowing for the sharing of physical channels between partitions. However, notwithstanding these recent advancements, the definition of partitions still requires the dedication of real hardware resource. As a consequence, logically partitioning a CEC will typically involve the purchase of more hardware than would be required to run the CEC in single image mode.

26.3.2 Software Partitioning

Software partitioning is accomplished through the installation and use of the VM/ESA operating system. VM/ESA manages the real hardware resources of a CEC, and makes it available to software defined virtual machines. The user of the VM/ESA operating system is provided with a virtual machine at logon time, and accesses all real hardware resources through that virtual machine. VM/ESA provides each virtual machine with storage, CPU resources, and peripheral devices. Often these resources are virtual (hence the name of the operating system). This means that peripheral devices, for example, are software constructs that emulate the real peripheral devices, and provide access to real underlying devices without dedicating that device to a particular virtual machine. The VM/ESA operating system manages the real devices such that the integrity of the devices is maintained, and individual virtual machines are prevented from accessing or destroying resources being used by another virtual machine.

Virtual machines are defined in a flat file known as the VM Directory. Each virtual machine is represented by a stanza within this file. A virtual machine definition consists of a name for the virtual machine (user ID) along with statements that define how many virtual CPUs are allocated, how much virtual storage is allocated, what virtual devices are allocated and their characteristics, along with privileges given to the virtual machine. Since the resources of a virtual machine are not real, the number of virtual machines that can be defined and active at any one time is not limited to the physical resources available. Using the example presented above for logical partitioning, it would be possible to log on all five of the virtual machines defined (each with 500MB of storage defined), even though the real CEC only has 1GB of central storage available. VM/ESA manages the use of real memory by paging real frames of central storage to slots on DASD based upon the use and reference patterns of the virtual machines.

Since software partitioning involves an operating system, some real hardware resources will need to be allocated to support this operating system. Thus the real resources of the CEC will now be distributed among three operating systems in our migration scenario instead of just two. It is possible however, to minimize the impact that VM/ESA has upon use of the real hardware resources. For example, instead of having VM/ESA create virtual DASD devices for a particular virtual machine, it is possible to dedicate certain real DASD devices to a virtual machine. This removes the simulation overhead incurred by having VM/ESA maintain a virtual device, and allows the virtual machine complete control over the particular DASD device. While the DASD device is dedicated to a particular virtual machine it is unavailable for use by any other virtual machine. It is also possible to dedicate central storage to a particular virtual machine. When this is done, it avoids the overhead of having VM/ESA perform paging operations, and allows the guest to occupy and manage a portion of real

central storage as if it were running natively on the processor. While storage is dedicated to a particular virtual machine, it is unavailable to other virtual machines, and also VM/ESA itself.

VM/ESA however, provides much more than simply hypervisor, and virtualization functions. It also provides a full function, interactive, virtual machine operating system, CMS. CMS provides an unmatched interactive environment for program execution, and program development. In addition, VM/ESA provides communication methods that allow virtual machines to communicate with one another, as well as a shared spool for use by virtual machines. These capabilities go far beyond the hypervisor and hardware management functions provided by PR/SM, setting VM/ESA apart among operating systems.

Since the migration from VSE to OS/390 will involve the creation of many operating system images to support various facets of the migration process, speed and flexibility in defining these images is essential. In addition, for these operating system images to be useful, they cannot operate as islands, but rather must be interconnected in as many ways as possible. For example, the images may need to share DASD (even though this is very difficult for VSE and OS/390 to accomplish), may need to have VTAM connections to support NJE, file transfer, and logon sessions, and may need shared access to printing or tape resources. Spending a lot of time on these details diminishes the focus that can be applied to the task at hand, namely migrating from VSE to OS/390. Therefore, it is essential to be able to quickly and effectively bind images together without extensive disruption to the entire operating environment.

26.3.3 Our Recommendation

With these characteristics of the migration in mind, and based upon the very brief introduction to the two partitioning environments, it is the recommendation of this document that customers use software partitioning with VM/ESA whenever possible. VM/ESA will provide the partitioning flexibility, and speed of implementation necessary to support the migration process. In addition, CMS and other products such as RSCS, TCP/IP, and VTAM that are available to run natively on VM/ESA can further enhance the migration process beyond what would be possible in a PR/SM only environment.

Many VSE customers today have VM/ESA installed, and are using it as a production hypervisor, or a hypervisor for test virtual machines. For these customers, this recommendation is business as usual. For VSE customers who are not currently using VM/ESA, this recommendation may seem to pose more of a problem. However, before rejecting this idea out of hand as too costly or difficult to implement, consider that recent pricing actions make ownership of VM/ESA along with another S/390 operating system very attractive and affordable. In addition, improvements in VM/ESA installation automation make it very easy to install, typically taking no more than a couple of hours. Factor in also the potential savings associated with not having to buy some additional hardware to support more LPARs (such as channels, control units for consoles and so on), and this suggestion becomes very attractive. In addition, consider that the introduction of VM/ESA can be useful to your enterprise far beyond the migration process. VM/ESA can continue to be used to provide a test environment for OS/390 guests used in the maintenance and test process. As your enterprise grows, and you begin to explore functions such as parallel sysplex in the OS/390 environment, VM/ESA′s virtual coupling facility support can enable you to define and run a sysplex under a single VM image without any real coupling facilities being defined, or coupling links being purchased and

installed. This support exists in VM/ESA Version 2 Release 3 on Multiprise 2000, 9672 G3, and G4 processors.

If you are currently using VM/ESA as a hypervisor for your production VSE guest(s), as well as for test VSE guests, then proceeding with the migration process involves nothing more than defining additional guest virtual machines for OS/390 images. In this environment, you no doubt have already developed expertise in making sure that your production VSE images are not impacted by the performance characteristics of your test VSE images. You will want to apply that discipline also to the OS/390 guests that are installed, and begin execution on your system. If you want to limit the resource consumption of your OS/390 guests, you can do so through the SHARE CP command, or SHARE directory entry along with the HARDLIMIT operand. In addition, you will want to ensure that your physical system has the resources to support the additional guest workload. For example, you will want to review your current utilization of central storage (what is the paging load on the VM system), CPU resources (what is the CPU busy percent), and available DASD. You will also want to look at the utilization of paging areas on DASD, and spool space.

26.3.3.1 Shared DASD

To provide the most flexibility in sharing DASD, you may want to consider defining the OS/390 DASD devices as full pack minidisks rather than dedicated devices. This would allow for sharing among OS/390 images with VM/ESA′s virtual reserve/release support, as well as for controlled sharing with VSE images. In fact, when sharing with VSE images, VM/ESA provides more protection that could easily be achieved in a native environment, by allowing for R/O links to be defined. One image can have a link defined to the minidisk as a R/O link, and the other can have the minidisk in R/W mode. For the image with the R/O link, the device appears to have the read inhibit switch set. There is no need to perform any manual activity within the guests, since if the guest having the R/O link attempts to write to the device it will be prevented by CP from doing so. Later, when the amount of sharing diminishes, and the need for better performance arises in the OS/390 guest, the devices can be dedicated to the OS/390 guest instead of accessed as full pack minidisks.

26.3.3.2 New Users of VM

If you do not currently use VM/ESA in your VSE environment, the introduction of VM/ESA will take more planning. Most likely you are currently running your VSE production images in separate LPARs, and have one or more test LPARs defined. The first choice you have to make is whether to continue running your CEC in LPAR mode, or run it in native mode with VM/ESA acting as the hypervisor for all of your VSE and OS/390 guests. The other choice you have is to continue to run your production workload in LPARs, and run VM/ESA along with the test VSE guests, and OS/390 guests in another partition. Given that the production environment is most likely well established in an LPAR mode, this latter suggestion would be the least disruptive to implement. The only caveat to be aware of with this approach is that it is not possible to run high performance preferred guests under VM/ESA when it is running in an LPAR. This however should not be a big impact since the use of guests under VM/ESA in this scenario is simply for testing. With this approach you would plan to run your production OS/390 image in an LPAR as is done currently with the production VSE image.

To be most effective, you would want to establish communication connections between the production VSE LPAR(s) and the VM/ESA LPAR, and then let

VM/ESA distribute that communication capability among the guest images using virtual channel to channel devices. Similarly, any DASD that is shared between the VSE LPAR(s) and the VM/ESA LPAR can be defined through R/O minidisk definitions owned by a place holder virtual machine, and accessed through R/O links from the OS/390 guests.

26.3.3.3 The Advantages of Guest Support in VM/ESA

You can use *Guest Support* in VM/ESA to develop, maintain, manage, and migrate *other operating systems* that make use of one of IBM′s 370, 370-XA, or ESA architectures. System programmers and application programmers often solve the problems they encounter by using the solutions already integrated or implicit in VM/ESA. Some of the reasons customers use VM/ESA Guest Support to run their VSE and/or MVS(OS/390) systems are:

- System Simulation
- Performance benefits
- Reduced hardware and migration cost
- Operations management
- Recovery management
- Interactive Computing, Application Development and Support
- Interactive program development tools
- Debug and trace tools
- Interactive data analysis and reduction
- Access to VM/ESA CMS applications
- Server consolidation
- DB2 guest sharing
- 3990 models 3 and 6 Fast Write Transparency
- Multiple 3270 Session Support

System Simulation

Every computer user has a requirement for a spare system to migrate/upgrade to a new release, try out a new idea or to isolate, debug and test fixes or develop and test new production applications. VM/ESA makes virtually unlimited “spare systems” available in virtual machines, each simulating a real machine with very high fidelity. The advantages of making one system look like many need hardly be explained here, but in the present context it is worth mentioning some of them:

- With VM/ESA′s Guest Support Virtual Machine(s) you can create an exact replica of your production system on which you can test your new programs, services, and procedures. VM/ESA′s Guest Support Virtual Machine is an extremely cost effective way to have your own test system (or as many as you′d like). It is also a safe way to test new function because with VM/ESA′s Guest Support, your test system(s) are contained within a guest virtual machine and securely isolated from your real production system, applications, and data.
- Guest Support gives you the flexibility to create a migration plan that fits the needs and schedules of your business. This allows your production systems to remain up during normal business hours, eliminates the work required to schedule, bring down and restore the production system when off-shift work is required. Also, the need to schedule and pay for off-shift programming is eliminated.
- You can create a multiple production system environment. For example, your installation might run VSE applications, be migrating to OS/390, and have a

new vendor package that runs under AIX. As guests of VM/ESA, all three can run efficiently while sharing one processor.

- You have production applications that need to be reworked to comply with Year 2000. With VM/ESA′s Guest Support you can bring up a duplicate of your production system, set the clock to a date and time beyond the year 2000 then perform test and application debugging without disrupting your current production system.
- You have production applications that need to be reworked to comply with changes required by the European Common Currency. With VM/ESA′s Guest Support you can bring up a duplicate of your production system, set the specific elements that affect currency and/or exchange rates, then perform test and application debugging without disrupting your current production system.
- If you are considering using Parallel Sysplex. VM/ESA supports guest coupling simulation on the IBM 9672 Parallel Enterprise Servers Generation 3 and Generation 4 and on the IBM Multiprise 2000 Servers (at the appropriate engineering change levels). VM/ESA Guest Coupling Simulation provides for the simulation of one or more complete parallel sysplexes within a single VM/ESA system image. The intent is to provide a pre-production testing platform for a coupled-system installation. Other than the processors required, there is no special hardware needed: no coupling links and no external coupling facilities. All guest operating systems coupled within a simulated sysplex can only be coupled (through simulated coupling links) to coupling facilities also running as guests of the same VM/ESA system. Up to 32 virtual machines can be coupled within a simulated sysplex, with each such virtual machine coupled to up to eight coupling facility virtual machines.

Performance Benefits

Guest systems may see performance improvements by exploiting VM/ESA features. For example, both virtual disk in storage and minidisk cache allow guests to avoid real I/Os by using data in storage and caching techniques.

Reduced Hardware and Migration Cost

Guest systems such as OS/390, MVS, TPF, VSE, VM and others can share devices such as channels, printers, and DASD, which VM/ESA efficiently manages. VM/ESA adds value to such devices merely by the way it manages them. A good example is VM/ESA′s minidisk support, which allows one real disk to function as if it were several smaller disks (such as multiple IPL-able minidisks). VM/ESA also simulates some hardware devices (such as unit record devices and CTC adapters).

For migrating to a new release from an older VM, VSE, MVS, OS/390, or TPF, VM/ESA gives you the ability to bring up the new system on the same physical processor saving you the cost of a separate processor or LPAR hardware. This new system can share the devices and resources of your existing VM/ESA system thus eliminating the cost of separate hardware for new system migration testing. When testing is complete switching over to your new production system is only a matter of configuration/table changes and can be accomplished in minutes. These technical and cost saving advantages provided by VM/ESA and VM′s Guest Support are fundamental requirements upon which the VM/ESA product was built and have been carefully refined over the years. This gives

VM/ESA and its customers exclusive technical advantages not available in any other operating system platform.

Operations Management

With PROP (VM/ESA′s PRogrammable OPerator) you can cut your console traffic substantially. This saves time and reduces errors.

Recovery Management

You can build guest virtual machines to simulate systems at your organization′s other sites. So, VM/ESA can become a disaster-recovery backup site.

Interactive Computing, Application Development and Support

Although it is not normally considered a guest function, VM/ESA′s CMS is the development and interactive platform-of-choice for many IBM customers both S/390 and non-S/390. One reason for this is VM/ESA offers many practical application development and support tools that make the job easy.

Interactive Program Development Tools

CMS, XEDIT, and REXX, NetREXX, Pipelines and others provide elegant, powerful, and convenient services to help you write programs.

Debug and Trace Tools

Debugging under CMS is much easier than in a batch environment, because you can see the results right away and make changes easily. With the advent of INSPECT and its truly interactive symbolic debugging, you can examine your code, executing one instruction at a time, making any necessary changes.

VM/ESA supports IBM′s Language Environment (LE), VisualAge, VisualLift and a host of others products related to S/390 based application development and support. These tools supported by VM/ESA speed up application development and support and are a strong complement to your overall business.

Interactive Data Analysis and Reduction

CMS Pipelines, REXX, XEDIT, and DB2 constitute a powerful toolbox for reducing, distributing, and analyzing data originating from the VSE, OS/390, TPF, and other systems running as a guest on VM/ESA.

Access to VM/ESA CMS Applications

VM/ESA supports many applications, such as several Web Server Products, OfficeVision, CMS OpenEdition, DB2, TCP/IP with a wide variety of the TCP/IP applications, TCP/IP Sockets APIs available in products such as REXX and CMS Pipelines. These products and others are attractive to customers running production and/or test OS/390, VSE, TPF systems on VM/ESA.

Server Consolidation

VM/ESA tools and functions and VM/ESA′s Guest Support give customers the ability to consolidate many diverse distributed systems into a single image. VM/ESA does this at a fraction of the cost required to maintain and service many

distributed servers. VSE, OS/390, TPF, AIX and others are discovering the enormous value VM/ESA brings to the table.

DB2 Guest Sharing

With DB2, VM, VSE and OS/390 users can now share a DB2 database among several VSE and/or OS/390 guests. For most customers, consolidating several guest databases into one DB2 database reduces administrative work, simplifies operation, increases data integrity, and improves performance. DB2 guest sharing also streamlines access to operational data by decision support personnel, who often use CMS-based tools. Furthermore, sharing one DB2 database gives VSE, OS/390 and VM applications access to remote data by way of Distributed Relational Database Architecture (DRDA) support. Finally, the DB2 VM Data Spaces Support Feature offers even higher performance for users of DB2 Version 3 Release 3 through ESA/390 architecture and VM/ESA Data Spaces.

3990 Models 3 and 6 Fast Write Transparency

VM/ESA makes 3990 models 3 and 6 DASD Fast Write and Cache Fast Write functions available to guests, such as some VSE releases which support 3380/3390s but not the 3990 models 3 and 6 Extended Functions. This allows customers to benefit immediately from moving to the newer storage controllers.

Multiple 3270 Session Support

VM/VSE users often do several things simultaneously on different operating systems. Operators can manage consoles for multiple guest virtual machines, while programmers can move between CMS sessions and virtual test systems. Not surprisingly, users want to be able to run several sessions at each terminal simultaneously to simplify their work. VM/ESA Pass-Through Facility (PVM) supports several concurrent sessions per user with the ability to switch from session to session using a command or hot-key. This function is also available for VM/OS/390 users.

26.3.3.4 Use of CMS

Lastly, it is worth considering the roll of CMS in the migration environment. CMS along with XEDIT and the Shared File System (SFS) provide an excellent environment for managing and modifying the many objects that will need to be moved between the VSE system and OS/390 system. JCL members and other source objects can be moved from the VSE environment and placed into a hierarchical directory structure within SFS. These directories can be accessed by multiple CMS users with complete data integrity ensured. Copies of directories and their contents can easily be made to freeze modification levels for easy backout. Modified objects can then be moved into the OS/390 environment and placed into appropriate partitioned data sets. The CMS environment then becomes the location where modifications are being made. This ensures that nothing in the VSE environment is inadvertently changed wiping out the original contents. In addition it provides a clearing house to quickly see what has been moved to the OS/390 environment if it appears as though critical elements are missing. Lastly, it provides a central location where both old and new copies can be compared side-by-side for problem determination, and tools such as REXX and CMS Pipelines to automate the management and comparison tasks.

26.3.3.5 OS/390 Guest Considerations

The considerations for defining OS/390 guests are no different from those associated with defining VSE guests. From the VM/ESA point of view, the actions taken to maximize the performance of a VSE guest, would be the same as those taken to maximize an OS/390 guest. For example, specifying resource goals is done through the SHARE directory statement. Best performance is achieved by making the guest a preferred guest (V=R, or V=F), along with dedicating as many devices as possible. Other VM level scheduler controls such as setting STORBUF, or the DSPSLICE would be modified in a similar manner for both VSE and OS/390, since from the VM point of view both are virtual machines with long running units of work.

26.3.4 Summary

In summary, use of VM/ESA as a migration tool will enable you to focus more on the migration tasks, and less on tasks associated with creating a migration environment. Since this document is concerned with migrating from VSE to OS/390, we will now turn to more specifics concerning that task.

26.4 Parallel Activities

Throughout the above stages of migration, there will be many activities overlapping with one another, not the least of which are your daily production computing workload and conversion activities. The following considerations should also be factored into your choice and design of test system, including their configuration, availability and performance characteristics.

26.4.1.1 Overlapped Activities

The number of ″test″ OS/390 systems will determine what can be done simultaneously by different people, or by the same people without having to restore volumes or re-IPL.

For example, you can schedule times for the maintenance system to be used alternately by systems programmers for applying and testing maintenance, for operator training, and for testing different operating system configurations or options.

26.4.2 Synchronizing VSE Applications with OS/390 Versions

After you convert your applications to OS/390, you cannot freeze them on VSE. Any changes to the programs, JCL, data, and operating procedures in the VSE production environment must be replicated to OS/390. You need to schedule a replication or re-conversion of these applications to the OS/390 libraries after they are converted.

26.5 Building the Initial OS/390 Test System

Once you turn your initial OS/390 system over to application programmers for the real conversion activities, you need to create a second OS/390 system for testing. As part of this, you should give careful consideration to what is shared between the two OS/390 systems, and how isolated they are.

26.5.1 OS/390 Maintenance Environment

Early in the project a test SMP/E environment needs to be designed and built. This process involves ″cloning″ the OS/390 system libraries to provide a new target to apply OS/390 maintenance without compromising the OS/390 production environment. Since OS/390 was installed at the beginning of the project, the maintenance level could be well over a year old by switch-over time. It is highly recommended that a maintenance cycle be performed before switch-over. (Testing timeliness, and so on will dictate the best time for this.) The maintenance environment can be designed simply to provide for alternate resident volumes and be IPLed and tested in the production environment or preferably built in a test logical partition (LP) or virtual machine. (This will most likely be the implementation after switch-over.)

Once application testing starts on the OS/390 system, it becomes a “production” system to many. Any disruption to their testing environment will impact their conversion efforts.

There are many variations and considerations of maintenance environments and most are okay as long as the availability requirements are understood and met. The important criteria is that the test system provides for a simple process to apply maintenance in an emergency situation and during regular preventative maintenance cycles without causing a system outage. The Redbook *OS/390 Software Management Cookbook* SG24-4775, although somewhat out-dated, provides some excellent discussion on this topic.

26.5.2 OS/390 Test Logical Partition

There are many things to consider when building a test logical partition (LP) for the maintenance environment. For example, will you be sharing DASD, catalogs, system parameters, subsystems and so on, or will this be completely isolated and serve as a rescue system? Many things such as communications between the two environments will have already been addressed by the work done connecting VSE to OS/390. The more subsystems available to test in the test LP, the less likely of a system outage when IPLing the new maintenance into production. Or, put another way, the closer your test system is to your production environment and workload, the less likely you will be surprised by problems in production.

26.5.3 Maintaining Your OS/390 Libraries and SMP/E Zones

You must keep your OS/390 target libraries and SMP/E target zones synchronized to preserve the integrity of your system. When the target libraries are backed up, the associated target zones must be backed up as well. You should create a procedure where the first step backs up the volumes, and the second step backs up the SMP/E target zone.

The reverse is true for restore. Whenever you restore your target libraries, you should simultaneously restore your SMP/E target zone.

26.6 Shared DASD vs. Cloned DASD

The issue of whether to share DASD volumes and data sets between systems is decided on the basis of DASD space availability, need for multiple versions of a file, and the ability to manage updates between the two systems.

26.6.1 Shared DASD between OS/390 Test Systems (vs. Cloned DASD)

The decision to share data sets and volumes or to make copies of them for each OS/390 system should be thought out carefully. Many OS/390 data sets and some volumes can be shared between multiple systems as long as updates are serialized and good change control procedures are followed. The recommended approach is to put multiple OS/390 systems in the same sysplex and use GRS to guarantee serialization of these resources. The alternative is to ″clone″ or make copies of the volumes or data sets, but this obviously takes more DASD space.

Referring to Table 45 on page 403, there are some volumes that can be shared between active OS/390 systems, and others that should never be shared:

System Libraries Separate SYSRES volumes should be maintained for each logical OS/390 system.

Distribution Libraries Share, but only update from the maintenance system.

Catalogs The master catalog should be fairly static, contain only the necessary entries, with the rest in user catalogs. The master catalog can be shared but highly controlled as to who can update it. User catalogs can be shared between systems, use GRS or manual procedures to serialize updates.

Paging Data Sets Use by only one system at a time (re-use by different test system as long as they are not both active at once.)

Spool & Checkpoint JES2 can share the spool and checkpoint between multiple members if you are comfortable with multi-access spool. Otherwise, they should be dedicated to each OS/390 system. The backup OS/390 system should have its own spool and checkpoint.

Softcopy Library Share these between all OS/390 systems.

DFSMShsm ML1 Share if using HSM on both systems.

Storage/Work Share or isolate depending on how much space you want to reserve for the ″production″ system.

User Libraries You can share the user program and data libraries, but you must keep track of separate version levels with different data set names or different user catalogs and volumes.

A lot of these decisions whether to share volumes, data sets and work space depends on how much you want to isolate the systems and manage change control.

26.6.2 Shared DASD between VSE and OS/390 (vs. Cloned DASD)

As mentioned in the previous section, it is risky to share DASD between VSE and OS/390 because there is no mechanism such as GRS to guarantee serialization between the two systems. If you decide to share, you must strictly control updates from the two systems, or be prepared to restore volumes, files and catalogs should they become corrupted.

Part 6. Running Your OS/390 System

Chapter 27. Orienting ICCF Users to TSO/ISPF

There are many facets of VSE/ICCF that are done differently in OS/390. TSO along with ISPF and SDSF provide functions that were previously done using VSE/ICCF.

27.1 TSO/ISPF and SDSF

ISPF is a dialog manager and runs under TSO on OS/390. ISPF provides a powerful environment that can be used for both development activities along with job submission. SDSF extends this environment by providing facilities that allow for both job monitoring and job output viewing. These tools are normally used by system programmers, application programmers and production control personnel.

ISPF is actually several distinct ″features″ integrated together:

- Dialog Manager (DM) provides services for application developers to easily create and display applications, including Display Services, Variable Services, Table Services, and File Tailoring Services.
- Program Development Facility (PDF) provides utilities and services for application developers to create and maintain applications, including Edit, View, and Browse, a wide range of data set utilities, and foreground and batch compilers.
- Software Configuration and Library Manager (SCLM) provides a robust environment for controlling a software development environment and tools to manage the environment.
- ISPF Client/Server provides application developers with the ability to both incorporate the workstation into the development process and use it to run applications. Existing ISPF applications will run ′in GUI mode′ with no changes, and a set of distributed services are available to edit and build using workstation tools.
- VisualAge ISPF, a visual development solution utilizing the composition editor of IBM′s VisualAge technology, can be used to create new ISPF panels and modify existing ISPF panels for use on 3270 and GUI screens.
- An ISPF Application Server and ISPF Workstation Agent Applet allows legacy (and new) ISPF applications to be accessible from the World Wide Web.

Other functions provided by ISPF include:

- Ability to communicate with OS/390 through TSO commands, CLISTs or REXX EXECs.
- Ability to split the physical display screen into two or more logical screens (ISPF enables a maximum of 32). The logical screens are treated as though they are independent ISPF sessions. For example, one could edit two members of a partitioned data set, or view error messages in the output of a compile job while editing the source.
- Ability to use referral lists for data set selection from the View Entry, Edit Entry, and most of the Utilities panels. Reference lists are active lists of data sets and libraries that you have referenced in your ISPF session. You can

also build lists of personal data sets. Personal data set lists are a good way to group (by project, for example) those data sets that you use frequently.

- Ability to run foreground and batch processors such as Assembler H, VS COBOL, VS FORTRAN, PL/I optimizing compiler, Binder/Linkage editor, C/370, REXX/370, and C/C++ for MVS/ESA.
- Ability to test individual dialog elements and complete dialogs using ISPF′s Dialog Test option.
- Ability to keep statistics about each data set member including which user updated it, date it was created, date and time it was changed.
- Ability to see and work with the data sets which are allocated to your TSO user ID using the ISPF ISRDDN program. Although ISRDDN is considered a diagnostic tool, you may find it very useful in many situations, such as:
  - viewing allocations including data set characteristics,
  - editing, viewing, browsing allocated data sets,
  - freeing allocations,
  - compressing allocated partitioned data sets,
  - querying ENQs against a data set,
  - locating where a member exists in a concatenation.

For more information about these features and functions see the *OS/390 ISPF User′s Guide* and *SDSF Guide and Reference*.

For the latest ISPF release features, hints and tips, free software, ISPF newsletters, and information on how to access ISPF forums, visit the ISPF Web site http://booksrv2.raleigh.ibm.com/ispf/

27.1.1 Editing Data Sets

While TSO provides an editor, it is rarely used; most editing of data sets is done using the ISPF editor. The ISPF editor edits both sequential and partitioned data sets, with the majority of activities centered around partitioned data sets. The system programmer can easily edit system data sets such as SYS1.PARMLIB, while development programmers can edit program source. Production control personnel can edit job and PROCedure data sets. The ISPF editor has a program interface so that the edit function is available from any ISPF dialog with a custom look.

Edit provides functions such as:

- locating a particular line in the data,
- submitting edit data as a job stream for background execution,
- setting RECOVERY mode on so that edit keeps track of any changes that you make while editing data and if a system crash occurs, you will be able to recover and continue editing from the last interaction,
- saving the data without ending the edit session,
- canceling edit without saving the data,
- using the COMPARE command to compare, display, and merge differences between the data being edited and another file,

- finding specific character strings in the data, changing them to other character strings or to exclude the lines that contain strings,
- setting HEX mode on to allow you to display data in hexadecimal format,
- using language sensitive coloring, which highlights program constructs based on the programming language, improving readability,
- using the HILITE command to change the enhanced coloring and language sensitive coloring options of the editor.

For situations where some type of repetitive change may be required, users can write their own edit macros to perform the needed changes. ISPF provides a macro language which allows users to perform editing functions from a REXX exec, CLIST, or program.

ISPF also provides online models that can be inserted into the dialog. A model is an example of a service call, panel format, table format, or message that contains the proper syntax and all the available parameters for the programming language being used. Since these models are online, they can be called directly into the member being edited.

Distributed edit provides a seamless interface to edit a host file using a workstation editor or a workstation file using the ISPF editor. Distributed edit offers a significant opportunity for offloading host CPU cycles.

For more information see *OS/390 ISPF Edit and Edit Macros.*

27.1.2 Submitting Jobs

TSO provides a SUBmit command that can be used from the TSO command line interface for submitting jobs. In addition one can submit jobs while editing a data set using the ISPF editor. If the first line of the data set is not a jobcard, then a jobcard will be automatically built using parameters from the TSO logon. In that case the job name would be the TSO user ID suffixed by a user supplied character. The RACF user ID for the job is normally the same RACF user ID as was used by the TSO logon.

27.1.3 Using ISPF Utilities

In addition to editing of data sets, ISPF utilities are available to allocate, delete, catalog, uncatalog, and compress data sets, and display statistics about an entire data set or volume. ISPF also allows you to copy, move, rename, print, delete, and display information about members in a partitioned data set.

Note: In some situations such as copying or compressing large partitioned data sets, it may be better to use a batch utility such as IEBCOPY run as part of a job, than to perform the function under TSO/ISPF.

ISPF provides a utility option to create the IDCAMS commands to define, delete, and list catalog information for VSAM data sets.

ISPF provides a SuperC utility to compare data sets of unlimited size and record length at the file, line, word, or byte level. There is also a Search-For utility that can be used to search your data sets or PDS members for one or more character strings.

27.1.4 Creating and Executing ISPF Applications

Since ISPF is a dialog manager, many other products have written dialogs, connect themselves through the main menu and run as additional ISPF functions. Products such as SDSF, IPCS, RACF, SMP/E and QMF all provide dialogs that run with ISPF.

Users or system programmers can also write their own dialogs for specific applications using ISPF′s DM services. They can use the display services to display information on a 3270 or GUI screen, variable services to share program variables between screen and program, table services to store persistent data across invocations of an application, and file tailoring services to format program data for output. ISPF screens can be built for the applications using the ISPF panel language syntax. If you are more familiar with an SGML type language you can create ISPF panel source using the ISPF Dialog Tag Language (DTL) and DTL compiler. ISPF for OS/390 R5 provides VisualAge ISPF which is a visual builder for ISPF panel source. This allows users the flexibility of creating or modifying ISPF panels without needing to know the syntax of the ISPF panel language or ISPF DTL.

In addition to display, table, variable, and file tailoring services, ISPF provides library services to perform tasks such as opening a data set, reading records, writing records, moving members, finding members, and retrieving member statistics.

The ISPF Client/Server provides the ability to run ISPF on a workstation and display the panels using the display function of the workstation operating system. ISPF clients are available for OS/2, Win 3.1, Win NT, AIX, HP/UX, and Solaris. When running in ISPF ′GUI mode′, the user has the option to display all non-fullscreen TSO data in an ISPF/TSO GUI window. This window is scrollable and it contains an input field for entering required user responses. The data in the window can be selected and copied to a file of his choice.

For more information see *OS/390 ISPF Dialog Developer′s Guide and Reference, OS/390 ISPF Services Guide, OS/390 ISPF Dialog Tag Language Guide and Reference,* and *OS/390 V2R5.0 ISPF Parts for VisualAge.*

Continuing with the strategy of extending ISPF′s capabilities to allow its customers to make use of emerging technologies, ISPF for OS/390 R5 added the ability to access ISPF applications from the Web. The ISPF Workstation Agent Applet starts automatically from a Web page, and the ISPF Application Server receives requests from the ISPF WSA Applet to start and run the application. This provides GUI display as the ISPF Client/Server; however, it uses standard Java display services and uses the ISPF Application Server to communicate with ISPF.

For more information see the *OS/390 V2R5.0 ISPF Application Server User′s Guide and Reference.*

27.1.5 Managing Projects

The SCLM component of ISPF provides source control for development and maintenance projects. The functions include locking a member when a user is editing it, promoting from one level to another, tracking who is changing it, and keeping copies of the changes. SCLM also provides configuration management, such as tracking included source, and compiling only the members that have changed or that have included members that have changed.

For more information see the *OS/390 ISPF Software Configuration and Library Manager Developer's Guide, OS/390 ISPF Software Configuration and Library Manager Project Manager's Guide,* and *OS/390 ISPF Software Configuration and Library Manager Reference.*

27.1.6 Tracking Jobs

SDSF allows you to monitor any job in the system or sysplex. Values such as CPU time and I/Os per second are displayed for all of the jobs running in the system. Sophisticated and easy-to-use sort and filter functions let users customize SDSF's display of jobs. In addition the TSO user who submitted a job can generally view any of the JES spool data sets for that job as they are being created. This includes any of the SYSOUT spool data sets in addition to the message spool data sets. SDSF allows users to control jobs with short commands. These "action characters" can be used, for example, to hold, release, or cancel one or more jobs. In addition if you are properly authorized you can view the SYSLOG and see the messages for all jobs. For more information see the *OS/390 SDSF Guide and Reference.*

27.1.7 Retrieving Output

SDSF provides the ability to view any JES spool data set that you are authorized to view. In most cases this would be the output from a job you submitted. Most of the output will be queued to a held output class such that it does not get sent to a printer. If after viewing the output the user determines that it does need to be printed, then the output can be re-queued using SDSF commands. JES spool data sets awaiting a printer can also be viewed. Using ISPF split screen for example, you could view a job message spool data set on one part of the split screen and view a SYSOUT spool data set on the other part.

ISPF also provides an option to directly retrieve (or view) JES spool data and store it in a sequential or partitioned data set, however this option is rarely used if SDSF is available.

27.1.8 Using SDSF for Operators

Some data centers may choose to use SDSF to run the OS/390 system since given the proper authority, all OS/390 operator commands can be entered through SDSF. This makes it very convenient since the operator using SDSF does not physically need to be located near the CPU. In this scenario, the operator would use SDSF to view the SYSLOG, entering commands, replying to WTORs as required. Other SDSF panels are available to show jobs awaiting execution, job executing, to allow display and control of such things as jobs awaiting execution, devices (printers, initiators, lines, and so on) and system resources (members of the MAS, nodes, and so on). It's important to give proper consideration to the security and levels of commands available through SDSF, such that application programmers are not allowed to enter normal operator commands. For more information see the *OS/390 SDSF Customization and Security* manual.

Chapter 28. Orientation to OS/390 Console Operation

28.1 Introduction

There are enough differences between VSE and OS/390 operations to warrant each operator and systems programmer attending a class on the subject.

This chapter is intended only to provide the reader with an overview of OS/390 console operations on a single system. (Multi-system or sysplex operation are not covered here.) It is not intended to replace more formal training which should be given to all OS/390 operators and system programmers. It is also not intended to replace the standard OS/390 Operations publications listed herein.

The OS/390 system can be operated without a lot of manual intervention if set up correctly, and various automation products are used.

28.1.1 Operating Hardware Consoles

Before you can operate the system, you must be able to configure the hardware elements, and initialize the processor using the hardware console which is usually integrated into the service processor of the CPU. These consoles vary from processor to processor, and most should be familiar to the VSE operator using the same hardware:

- Hardware Management Console (HMC)

  The IBM S/390 CMOS (9672) and Multiprise processors have hardware consoles which are used to power on, IPL, reset, configure, and power off the processor. (The HMC is not required for an S/390 Multiprise.)

- S/390 Multiprise Stand-Alone Support Element (SASE)

  This an external console running OS/2 Warp which can be used for hardware management and has a look and feel similar to the HMC.

- Other S/390 System Consoles

  The IBM ES/9000 9021, 9121, and 9221 processors have integrated system consoles using the Service Processor or Processor Control Element (PCE) to power on, IPL, reset, configure, and power off the processor.

All of these hardware system consoles have interfaces for operating the OS/390 software, but are not usually used except for emergency or test.

28.2 Understanding the Operator Interfaces

Operator commands come in many flavors (MVS, JES2, SDSF, other subsystem), and can be entered through many different interfaces, such as:

- The System Console
- Hardware Management Console (HMC)
- MCS (Multiple Console Support) Consoles
- Subsystem Consoles
- Extended MCS Consoles
- TSO Operator Consoles
- SDSF (System Display and Control Facility)

- NetView Consoles
- TSCF Consoles
- Programmed Operator Subsystems

This chapter will only deal with MCS and SDSF consoles.

28.2.1 Controlling Consoles

There is more to operating an OS/390 system than just entering commands and reading the messages. You should also be familiar with various console configuration options. See Chapters 2 and 3 in *MVS Commands* for a description of these console operations:

- Defining and Changing Console Characteristics
- Potential Effects of Altering Console Attributes
- Changing Console Characteristics
- Controlling System Messages and Commands
- Defining Program Function Keys (PFKs)
- Hardcopy Processing

28.2.2 Managing Display Consoles

For MCS consoles (those attached to non-SNA control units), you will need to understand how to use some of the basic CONTROL (alias K) command parameters when you first start operating an OS/390 system. Otherwise, you can easily get your console ″locked up″ by non-deletable messages or not scrollable.

Here are some examples of the basic ″K″ commands:

K - clear the screen.
K S - show the current settings so they can be over-typed.
K E,4 - delete the message on line 4.
K E,4,10 - delete the non-action messages on lines 4-10.
K E,D - delete a status display from a display area.
K A,NONE - get rid of display areas on this console.

Use the ″D C,K″ command to display the CONTROL command functions and ″D C,A″ to display the status of active consoles.

See Chapters 2 and 3 and Section 4.7 in *MVS Commands* for details. (Most of these do not apply to Extended MCS consoles.)

28.2.2.1 Console Modes

There are several ″modes″ by which MCS consoles can operate and by which messages are cleared from the screen. Type ″K S″ to see the current setting of the console you are sitting at, and then you can over-type what parameter values you want to change. Here is an example of ″Roll″ mode which is recommended for unattended console operation:

```
K S,DEL=R,SEG=28,CON=N,RNUM=14,RTME=001,MFORM=(T,J)
```

If there is always an operator present, you can use ″DEL=RD″ which rolls the deletable messages, but not those messages requiring action.

28.2.2.2 Display Areas

These may be handy for operating a console with a lot of traffic, when you want to see a multi-line display without having it roll off the screen. However, operators need to be able to manipulate these display areas for efficient operation.

Enter ″K A,NONE″ to get rid of display areas, or ″K A,nn″ to create an area of size ″nn″.

An alternative is to use the SDSF ULOG panel which limits the display to just those commands and messages issued for the specific user. It can also be scrolled back and forth to review past messages.

28.2.2.3 PFKeys

Program Function Keys (PFKs) can be set with the CONTROL command ″K n,PFK″ or by activating a PFK Table. You can display the definitions with ″D PFK″.

28.2.3 Extended MCS Consoles

Extended MCS consoles are sort of like ″virtual″ MCS (Multiple Console Support) consoles that are implemented through software.

You can define a TSO/E user to operate an extended MCS console from a TSO/E terminal. The user can issue the TSO/E CONSOLE command to activate the extended MCS console or use SDSF.

An installation can also write an application program to act as an extended MCS console. An authorized program issues the MVS authorized macro MCSOPER to activate and control the extended MCS console and uses other MVS macros and services to receive messages and send commands.

See *OS/390 MVS Planning: Operations*, GC28-1760 for details.

28.2.3.1 Using the TSO/E Functions

Use the CONSOLE command to establish a full extended MCS console mode in “Command” mode or “Conversational” mode. (The OPERATOR command can also be used for operator commands, but is limited to the OPERATOR subcommands.) See *OS/390 TSO/E System Programming Command Reference*, SC28-1972 for details.

28.2.3.2 Using SDSF for System Operation

Below is the Primary Option Menu for SDSF showing you the basic panels you can use as a full-screen system operator.

```
HQX1800 -----------------  SDSF PRIMARY OPTION MENU  -------------------------
COMMAND INPUT ===>                                            SCROLL ===> CSR

    LOG       - Display the system log
    DA        - Display active users in the sysplex
    I         - Display jobs in the JES2 input queue
    O         - Display jobs in the JES2 output queue
    H         - Display jobs in the JES2 held output queue
    ST        - Display status of jobs in the JES2 queues
    PR        - Display JES2 printers on this system
    PUN       - Display JES2 punches on this system
    RDR       - Display JES2 readers on this system
    INIT      - Display JES2 initiators on this system
    MAS       - Display JES2 members in the MAS
    LINE      - Display JES2 lines on this system
    NODE      - Display JES2 nodes on this system
    SO        - Display JES2 spool offload for this system
    ULOG      - Display user session log
    END       - Exit SDSF
```

Displaying the system log is handy because you can scroll back and forth and search for text strings. There are several panels to display jobs and output, which can show you the backlog of work for initiators, printers, and transmitters. There are also panels for JES2 devices such as printers, punches, readers, lines, nodes, members, spool offload devices, and other JES2 resources.

You can issue MVS and JES2 operator commands from the command line, and many panels support simple action characters and over-typeable fields for JES2 devices and parameters.

Each of these panels provide simple action commands (modeled after the JES2 command verbs) to control the work such as: B-Backspace, C-Cancel, D-Display, E-Restart, F-Forward, I-Interrupt, N-Repeat, P-Stop, S-Start, and Z-Halt.

SDSF has online HELP panels, a Tutorial, and can link to online (softcopy) books on BookManager READ/MVS for Messages.

28.2.4 Understanding Message Formats and Replies

See *OS/390 MVS System Messages, Volumes 1 to 5*, GC28-1784 thru GC28-1788 for detail message descriptions. The front of each book also describes the general format of MVS messages.

To reply to a WTOR (Write To Operator with Reply), you can either enter *R nn,'reply'* or use the short form and type the one or two-digit reply ID followed by the reply. For example, to reply ″U″ to a WTOR with the ID=07, you can either enter ″R 07,'U'″, or you can enter ″7u″.

The short form is not available when JES2 is not up unless you specify CON=NOJES3 in the IEASYSnn member of parmlib.

28.3 Controlling the OS/390 System

The OS/390 System commands are only a subset of the commands necessary to operate the system. JES2, VTAM, and many other OS/390 components also have a command language which is used to operate their subsystems. We recommend that systems programmers and operators attend a class on basic OS/390 (MVS) and JES2 facilities. In addition, there are more advanced classes such as the ″CMOS Complex Systems Availability and Recovery″ (five days).

This section is for a new operator controlling a single OS/390 system using MCS consoles with JES2 and SDSF.

28.3.1 Starting the System

Here is an overview of the steps to start OS/390:

1. Prepare the System Hardware (power on, IML, and configure)
2. Load the System Software (IPL-ing MVS)
3. Specify the System Parameters
4. Set the Time and Date (if required)
5. Start JES2 - ″S JES2,PARM=NOREQ″
6. Start VTAM - ″S NET,,,LIST=xxx″
7. Start TSO - ″S TSO″
8. Start RMF - ″S RMF″
9. Start other subsystems, such as Automation, CICS and so on

These commands can be automated in the COMMANDxx member of parmlib, or through some other automation product, so they do not have to be entered by a human operator.

28.3.2 Displaying System Status

Depending what you want to display, you can either enter an MVS command, a JES2 command, use SDSF, or use another subsystem display.

The MVS ″DISPLAY″ command (abbreviated ″D″) has many different objects and parameters. Here are some basic ″D″ commands:

```
D R,L - List all outstanding WTOR messages awaiting reply
D R,U - List all outstanding Mount Requests (for example, tape mounts)
D A,L - List Active Jobs, TSO users, and Started Tasks
D D,S - Display the Dump Options
D SMF - Display the SMF status
D LOGREC - Display the status of LOGREC
D TRACE - Display the status of TRACE
D C - Display Console configuration
```

There are many other Display commands and parameters. See section 4.9 in *MVS Commands* for the details.

28.3.3 Stopping the System

There are several ways to stop or halt the system, and important subsystems. Here is a simple example of commands to stop the system:

$Pxxx	Drain all active JES2 printers, initiators and so on
P TSO	Stop Time-sharing
Z NET,QUICK	Stop VTAM
C APPC	Stop APPC (if active)
D A,L	Display active jobs to see what else needs to be stopped
P RMF	Stop RMF (and any other active subsystems)
$PJES2	Stop JES2 (see options below)
Z EOD	Flush all SMF buffers, LOGREC, Caches and so on to external DASD

Now you can safely power the processor off or re-IPL.

Stopping JES2

There are various flavors of the $PJES2 (stop JES2) command:

$P JES2	Stop JES2 after ″all available functions are complete.″ This usually takes forever to drain all the devices, processes, jobs and started tasks, that were started under JES.
$P JES2,TERM	This is quick and the recommended way to stop JES2 if you know you are going to re-IPL.
$P JES2,ABEND	Stop JES2 immediately so you can ″hot-start″ it without an intervening IPL.

28.4 Controlling Devices

Some devices such as the system volumes or work packs are permanently resident, always mounted, and shared amongst users. Devices such as tape drives are allocated to one job at a time as they are needed. Others, like printers are managed by JES2 so they can be used by all jobs.

28.4.1 Displaying the Status of Devices

Use the ″D U″ command to see the status of various devices by device address or device type. For example, enter ″D U,DASD,ONLINE″ to display online DASD units and their volume serial numbers.

Use the ″D M″ command to see the paths to devices.

28.4.2 Understanding Device Allocation

Batch jobs and other users allocate devices for their work, and operators may be prompted to mount or otherwise respond to these requests.

See section 1.8 ″Interacting with System Functions″ in *MVS Commands* for a description of Device allocation, Hot I/O detection, and Device ″boxing″. Section 1.9 describes the SWAP command to respond to Failing Devices.

28.4.3 JES2 Devices

Devices such as printers, punches, TP lines, and spool offload tapes can be allocated by JES2 dynamically. The following JES2 command verbs are used to control JES2 devices and are followed by the device name such as PRT1, PUN(12), LINE(5).

$D Display the device
$S Start the device
$P Stop the device when it is through with the current job
$Z Halt the device immediately
$C Cancel the job currently being processed on the device
$T Change the device setup characteristics
$I Interrupt printer or punch
$N Repeat printer or punch
$E Restart the device (interrupt and re-queue)
$B Backspace printer or punch
$F Forward space printer or punch

PSF printers are defined as JES2 printers and controlled through the same commands used for JES2-controlled printers.

28.4.4 SDSF Device Panels

There are separate SDSF panels for JES2 devices such as printers, punches, readers, lines, remote workstations, and NJE nodes. These may be more convenient than the JES2 commands, because you can:

- Display many devices in a tabular format,
- Issue any operator command in a simpler form (complete with ″Help″),
- Change the characteristics of a device by over-typing fields,
- Manage input and output queues for these devices.

28.5 Controlling TSO Users, Jobs and Started Tasks

Time-sharing users, batch jobs, and started tasks all represent work being performed on the system and reside in their own address space. They are initiated in different ways, but all can be displayed and controlled in similar ways with MVS commands, JES2 commands and SDSF panels. However, there are subtle differences that make some better than others.

28.5.1 Displaying Work on Your System

TSO users, batch jobs, and started tasks each run in their own address space, and represent work by one or more users on your system. There are MVS and JES2 commands to display and control them. SDSF, RMF, and other monitors also have operator interfaces to monitor and control.

Each of these has different information and presentation. Try each of them and use what seems best for your purposes.

28.5.1.1 MVS Commands

Use the DISPLAY JOBS, J, A, or TS command to display information about current system activity, including time-sharing users, batch jobs, and started tasks. The MVS ″TRACK″ and ″MONITOR″ commands also provide assistance with periodic updated displays in a display area.

28.5.1.2 JES2 Commands

There are many JES2 commands to display the work on your system:

$DA Use the $DA command to display information about active jobs, started tasks, and time-sharing users. This also shows jobs active on JES2 devices such as printers.

$DN Use the $DN command with various filters (Q=, V=, R=) to display jobs in specific phases of JES2 processing, on specific spool volumes, or with specified route-codes.

$DJ Use the $DJ, $DS, or $DT commands to display information about jobs, started tasks, or time-sharing users, known to JES2 on any queue, active or not.

$DJQ The $DJOBQ command is even more powerful with many different filters including wild-card characters to display information about jobs known to JES2.

There are many JES2 commands to control this work such as $C (cancel), $H (hold), $T (modify), $P (purge), and $E (restart).

28.5.1.3 SDSF Panels

There are four basic panels in SDSF to show active jobs:

ST Status of all jobs known to JES2, along with many characteristics such as amount of spool space being used.

DA Display of all Active jobs, started tasks and time sharing users, along with ASID, and RMF information about CPU, storage, and I/O rates.

I Input queue display of all jobs waiting for execution. This a good way to see what your backlog is for initiators.

O Output queue display to show all job output elements waiting to be printed or punched, or transmitted to another or remote node.

H Held output display to show all output elements that are held waiting for TSO output.

Each of these panels provide simple action commands to control the work such as A=Release, C=Cancel, D=Display, E=Restart, H=Hold, I=Info, J=Start, L=List, O=Release, P=Purge, Q=Outdesc, S=Browse, and X=Print.

28.5.1.4 RMF and Other Monitors

There are other facilities to monitor your work in the system such as the RMF Monitor II Address Space Reports.

28.5.2 Controlling Time Sharing Users

TSO/E users logon through terminals controlled by VTAM. You can use MVS or JES2 commands to control TSO users and their output:

- Send a Message to a TSO User with the MVS ″SEND ′message_text′,U=userid″ command. Be careful not to omit the ″,U=( )″ operand, or your message will be sent to all TSO users and they will get aggravated with this if it happens repeatedly.
- Cancel the TSO session with the MVS ″CANCEL U=userid″ command, or the JES2 $C command.

  There may also be times when you cannot cancel a TSO user with the $C command, so you will have to force the address space down with the ″FORCE″ command. This may be necessary for a TSO user to get out of a ″hung″ condition, and log back on again.
- Release, Cancel, or Modify TSO held output with the JES2 $O or $TO command.

The SDSF DA, ST, O and H panels can also be used to control TSO users or their queued output. (Any JES2 command can be issued through these panels.)

28.5.3 Controlling Batch Jobs

Batch jobs are submitted by TSO users, or by other programs such as batch job scheduling systems like OPC/A. They are queued on the ″Job Queue″ by JES2 and selected by batch initiators according to your installation′s job scheduling criteria. (WLM or JES can manage the initiators.)

Jobs can also be started by the operator from the console with the MVS START command, but then they behave very much like ″started tasks″. (See the next topic below.)

Jobs can be canceled by the operator with either an MVS Cancel command, or the JES2 $C command.

The SDSF DA, ST, O and H panels can also be used to control batch jobs or their queued output.

28.5.4 Controlling Started Tasks

Started Tasks (or ″STCs″) are like batch jobs, but started by the MVS ″START″ command (abbreviated ″S″) from an operator console instead of submitted by a TSO user or another job. You can override many parameters on the START command, including the name that shows up on a display.

Other commands used to control started tasks are described in *OS/390 MVS System Commands*, GC28-1781.

- Display status about the started task with the MVS ″DISPLAY″ command or JES2 $D command.
- Modify the started task with the MVS ″MODIFY″ command.
- Stop the started task with the MVS ″STOP″ command.
- Cancel the started task with the MVS ″CANCEL″ or JES2 $C command.

  If they fail to stop or cannot be canceled, they can be ″Forced″ with the MVS ″FORCE″ command.

28.6 Managing Remote Operations

As with VSE, remote systems and workstations can communicate through NJE or RJE via commands and messages. Some remote workstations and systems have console operators, while others may not. The remainder of this chapter describes how to communicate via JES2 commands.

28.6.1 JES2 RJE Operations

There are many kinds of remote workstations: BSC and SNA, ones with just a reader and printer, and others with consoles, disks and spooling capability. Some have human operators, others do not. Some must be managed by the central host operators, others by a remote operator.

28.6.1.1 Host Operations

Usually, the remote operator controls the RJE session once the lines and interfaces are enabled. However, the host OS/390 operator can also initiate some sessions and may become involved with recovery operations if problems arise. Here are some JES2 commands to support RJE:

$S LGNn Start the JES2/VTAM ACB (for SNA remotes)

$S LINE(nn) Start the line

$S RMT(nn) Start the RJE session (SNA only)

$E LINE(nn) Re-start the line

$P LINE(nn) Drain the line

$D Mnn,'Please drain your printer'
Send a message to the remote operator.

28.6.1.2 Remote Workstation Operations

You should set up an RJE workstation (of each type) to orient your remote operators to the JES2 environment, show them how to submit jobs, retrieve output, and enter commands.

Here are some JES2 RJE statements to sign on and off:

- For BSC Remotes, use **/*SIGNON** to sign on, and **/*SIGNOFF** to sign off.
- For SNA Remotes, use **LOGON** - sign on to JES2, and **LOGOFF** - sign off (these are actually VTAM commands).

See Chapter 6 in the *JES2 Initialization and Tuning Guide* for more details.

The most common commands for RJE are for printers:

$S PRn Start the printer initially or after a setup message

$DF Display forms queued to this remote

$D JOBQ,CMDAUTH=R3
Display jobs that can be affected by Remote 3

$DO JOBQ,DEST=R3
Display output that is destined for Remote 3

$T PRn,F=xxxx,C=...
Set up printer n for other forms, classes and so on

$D MMn,′Please restart my printer′
Send a message to the operator on member n

See *JES2 Commands* for details. The “Remote Job Entry” section in Chapter 2 has good guidance information for RJE operations, and Chapter 5 has detailed command syntax descriptions.

Remotes Without Consoles

If you don′t have a console on your remote workstation, you can still submit commands and JES2 control cards through the logical (or physical) card reader.

JES2 Control Cards

The following JES2 control cards can be placed within a jobstream to communicate with the host system or central operator. (They can also be used by anyone submitting a job, subject to installation restrictions.)

/*$command Enter a JES2 operator command (for example, **/*$SPRT1**)

/*MESSAGE Send the message to the operator console

/*NOTIFY Send notification messages to specified users

/*SETUP Hold the job for specified volumes to be mounted

Other JES2 control cards such as /*JOBPARM, /*ROUTE, and /*PRIORITY can also be used to control specific jobs. See *OS/390 MVS JCL Reference*, GC28-1757 for details.

Command Authority for Remote Operators

In general, a remote operator can only display and control jobs which are submitted or “owned” by that remote system, and devices that are attached to that remote system.

The table titled “Remote Entry Restrictions” in Chapter 2 of *JES2 Commands* describes the RJE and NJE authority required for all JES2 commands.

28.6.1.3 Using SDSF Panels for RJE

The Line panel in SDSF allow you to monitor and manage RJE lines and devices. The Printer, Punch, and Reader panels also show remote (RJE) devices and can be configured for remote operators.

28.6.2 NJE Operations

Host operators on each node can display and manage NJE lines, sessions, connections and paths to other nodes. They can also send commands to display the configuration on other nodes. Here are some JES2 commands to support NJE:

$S LGNn Start the JES2/VTAM ACB (for SNA remotes)

$S LINE(nn) Start the line

$S N,NODE=node_name
Start the NJE session

$D NODE(node_name)
Display the status of a node

$D PATH(node_name)
Display the path(s) to another node

$D Nxx.′$D NODE(yy)′
Send a command to node xx to display the status of node yy

$D MNn,′Please drain your session′
Send a message to an operator on another node

See *JES2 Commands* for details. Chapter 4 has good guidance information for NJE operations, and Chapter 5 has detailed command syntax descriptions.

Command Authority for Remote Operators

An operator on one node can send commands to another node, subject to the authorization established by that remote node. In general, an operator is limited to a subset of commands that display and control jobs which are submitted or "owned" by that node, and devices that are attached to that node. In JES2, this is controlled by the AUTH parameter on the NODE initialization statement.

See the table titled "Remote Entry Restrictions" in Chapter 2 of *JES2 Commands* which describes the authority required for all JES2 commands.

28.6.2.1 Using SDSF Panels for NJE

The Line and Node panels of SDSF allow you to monitor and manage NJE activity.

See the *SDSF Guide and Reference* and *SDSF Customization and Security* for details.

Chapter 29. Orientation for Utilities

29.1 IEBxxx or IEHxxx

There are many utilities in OS/390 provided by DFSMS/MVS to assist you in organizing and maintaining data (most of them start with ″IEB″ or ″IEH″). These are simple programs which perform commonly needed functions. See ″Guide to Utility Program Functions″ in Chapter 1 of *DFSMS/MVS Utilities*, SC26-4926.

29.2 IEBCOPY

IEBCOPY is a utility program used to make copies of, and to maintain, partitioned data sets. In addition to the copy function, IEBCOPY performs the following maintenance operations:

- Compression - the members of a partitioned data set are moved together (compressed) to eliminate the unused space that results from changing existing members.
- Merge - two or more partitioned data sets are merged into a third data set.

Information on IEBCOPY is provided in *DFSMS/MVS Utilities*, SC26-4926.

29.3 IDCAMS

IDCAMS (Access Method Services) is a utility program that is used to manipulate all access methods except partitioned. IDCAMS is recommended for use with SMS managed data sets. This utility reads control statements and performs data set functions such as creation, deletion, cataloging, and uncataloging. In addition, IDCAMS performs the backup and restore functions for VSAM data sets. Information on IDCAMS is provided in the following publications:

- *DFSMS/MVS Access Methods Services for ICF* SC26-4906

  Describes IDCAMS commands for using integrated catalog facility catalogs.
- *DFSMS/MVS Access Method Services for VSAM*, SC26-4905

  Describes IDCAMS commands for using VSAM catalogs.
- *DFSMS/MVS Summary of Access Method Services for ICF*, SX26-3807.

  Provides a summary of IDCAMS commands for integrated catalog facility catalogs.

29.4 IEBGENER

The main use of this utility is for moving data. You can use IEBGENER to:

- Create a backup copy of a sequential data set, or a member of a partitioned data set or PDSE.
- Produce a partitioned data set or PDSE, or a member of a partitioned data set or PDSE, from a sequential data set.
- Expand an existing partitioned data set or PDSE by creating partitioned members and merging them into the existing data set.
- Manipulate data sets containing double-byte character set data.
- Print sequential data sets or members of partitioned data sets or PDSEs.

- Reblock or change the logical record length of a data set.
- Copy user labels on sequential output data sets.
- Supply editing facilities and exits for your routines that process labels, manipulate input data, create keys, and handle permanent input/output errors.

Information on IEBGENER is provided in *DFSMS/MVS Utilities*, SC26-4926.

29.5 DFSMSdss

DFSMSdss is a direct access storage device (DASD) data and space management tool. DFSMSdss works on DASD volumes only in the MVS environment. You can use DFSMSdss to:

- Copy and move data sets between volumes of like and unlike device types
- Dump and restore data sets, entire volumes, or specific tracks
- Convert data sets and volumes to and from SMS management
- Compress partitioned data sets
- Release unused space in data sets
- Reduce or eliminate DASD free-space fragmentation by consolidating free space on a volume

Information on DFSMSdss is provided in the following publications:

- *DFSMS/MVS DFSMSdss Storage Administration Guide*, SC26-4930.

  Describes DFSMSdss tasks for copying, dumping, and restoring data sets and DASD volumes.
- *DFSMSdss Storage Administration Reference*, SC26-4929.

  Describes DFSMSdss commands, including syntax and coding examples.

Chapter 30. Systems Management Philosophy and Methodology

Many VSE installations have small staff and have mature systems which are changed relatively infrequently. As a user migrates from VSE to OS/390, their entire system -- hardware, software, and connections -- will be subject to more frequent changes. The workloads supported by their system will grow in complexity and criticality to their business. At the same time, many of the tools and methods for managing the VSE environment will become obsolete, and new techniques and tools will have to be learned and implemented. In this chapter, we will discuss the opportunities to implement more formal system management procedures, the IBM products that can support these procedures, and the benefits that OS/390 users receive from the implementation of these procedures.

It is not the intent of this chapter to describe specific tools, methodologies, or services in great technical detail. The practice of system management disciplines can provide great benefits of productivity and time savings during a migration, especially as all personnel involved are working with new tools and in new environments.

The VSE/SP and VSE/ESA systems and MVS and OS/390 systems have some conceptual similarities, but due to the scope of changes that must be made during a migration, the migration project is an ideal opportunity to introduce more formal system management disciplines. The following specific disciplines will be discussed below:

1. 30.1, The Philosophy of Systems Management
2. 30.2, Change Management
3. 30.3, Problem Management
4. 30.4, Performance Management
5. 30.5, Operations Management
6. 30.6, Security Management
7. 30.7, Configuration Management
8. 30.8, Asset Management
9. 30.9, Accounting Management

These topics will be discussed in order within this chapter.

30.1 The Philosophy of Systems Management

30.1.1 Systems Management Overview

System management is responsible for delivering effective and efficient information technology services, and becomes more critical as the number of components, workloads, changes, and overall complexity increases.

Systems Management disciplines are aptly named -- through exercise of these disciplines, we learn to manage our system in a better manner -- with the result being a system better able to service the needs of all the different classes of users of the computer system.

In many smaller computer installations, where at most a few people are involved in system changes, ad-hoc and informal techniques have often proved adequate. All or us have seen a case where the complaint is lodged: ″It doesn′t work right.″ Oh, what seems to be the problem? ″Payroll doesn′t work right.″ ″What did you change?″ Nothing...

As systems become more complex, the staff working with the systems becomes larger, the number of components becomes greater, the location of the components becomes more dispersed, and the complexity of the connectivity between components increases. It becomes increasingly difficult to know what is really causing problems, what changes have been made and where, even what components are in use and what their state is at any given time.

A structured or organized approach to systems management activities is required. Over the past 15 years, several groups, including IBM, have defined the systems management tasks in terms of disciplines. Examples of these efforts include:

- IBM′s Information Systems Management Architecture (ISMA)
- Guide/Share User Group definitions
- OSI′s System Management Functional Areas (SMFA)
- IBM′s SystemView
- IBM′s Information Technology Process Model (ITPM), which updates ISMA
- IBM′s TME 10, which is an update to SystemView

In practice, the objective is not to adhere to a defined set of disciplines, but rather to follow a structured approach in defining the tasks to be achieved and the disciplines required to achieve them.

As one example, when major changes in the OS/390 environment and infrastructure are made, it will be very difficult to actually know what changes have been made and their impact since something last worked correctly, unless a change management discipline has been adopted. This is especially critical during the migration, since a large volume of changes is occurring.

Change management provides a standard way of introducing changes into the computer system with much less risk or system outage on one hand and with quicker resolution of system outages should they occur on the other. To achieve these benefits, those who have the skills and the authority to make changes to the system configuration, whether hardware or software, systems or application programming, give up the freedom to make changes whenever they desire and however they desire in order to improve the stability of the overall system.

The discussions justifying other systems management disciplines follow similar logic, and will not be explicitly stated in the sections below where the disciplines are described. The benefits of this approach are that it:

- Ensures that the wide range of activities required are covered.
- Ensures a complete solution.
- Allows a common process to cross physical and logical resource boundaries: that is, to have one problem management system, one change management system, and so on.

- Places a stronger emphasis on service, as it promotes keeping one′s ″eye on the ball.″
- Provides more effective and productive processes.

Challenges in this approach are not just to segment the activities, but also to recognize how the disciplines interrelate and how they cross functional boundaries. Also, all responsibilities must be assigned and understood, and disciplines documented.

30.1.2 Systems Management Scope - What Needs to be Managed?

In almost all systems, hardware (at the mainframe site, anyway) is reasonably controlled. As our systems become networks of autonomous users who can control their own configuration and setup, it is possible for an end user to destroy his or her own ability to interact with the host and other server systems. In some cases, they can also impact the connections and services to other users.

Host operating system and subsystem software (0S/390, ACF/VTAM, CICS/ESA, DB2) similarly have a relatively small group of people who control and manage their attributes and status. In the new OS/390 environment, however, it is likely that the number of people responsible for these components could be larger than is typical in computer systems running VSE/ESA or VM/VSE. In addition, problems and changes in one component may affect other components and workloads. Because of the increase in number of people who control these components, it is recommended to implement Systems Management in a way that all the systems support personnel will be aware of planned and completed management activities. In addition, this process permits the implementation of peer review of planned activities, which both catches errors and educates other staff members in the various Systems Management disciplines.

Application program software is likely to be impacted with higher volumes during an 0S/390 migration in several areas. For one example, it is likely that application source programs will be changed (from DOS/VS COBOL to COBOL for MVS, and VSE assembler macros to OS/390 macros). Much of this will be automated, but because of new language nuances, program maintenance and development will require more effort, at least to begin with. Systems Management disciplines such as change and operations management can help reduce this extra effort by avoiding rediscovery, identifying reasons for failure, and more. Change management will help control JCL changes, application setup and run instructions, and more. Operations management will invoke and monitor the applications, provide for problem bypass, and provide feedback for any further changes to be made.

Overall, then, you should realize more benefits by extending the scope of your implementation of systems management disciplines all the way from hardware configuration and setup on one end to application programming and JCL on the other end. Whether the discipline being applied is Change Management, Problem Management, Performance Management, Operations Management, Security Management, or others, the most benefit can be derived when the scope for the discipline covers the system.

The disciplines should also include the network elements (we will define network as anything on the non-mainframe side of a communications controller); the availability of the workloads to the users will depend upon those elements as

well, and the disciplines should include those to avoid ″the hole in the boat isn′t on my side of the ship, so all is well″ syndrome that can develop.

30.1.3 The Role of Automation

Automation involves the ability to correct, bypass, or circumvent failed system and network elements and applications, based on defined policies and using hardware or software functions without human intervention. Automation improves availability and reduces operational costs.

To put it bluntly, as your environment grows, it will be impossible to manage it without automation. The number of systems, subsystems, applications, connections, files, databases and so on that have to be monitored and controlled will simply be overwhelming for any size staff (if you can afford and even find a large enough staff). Automation can be applied to all of the disciplines to make them more efficient; to do this it must follow prescribed management policies, which can be developed directly from a structured approach.

The following sections describe a set of the typical categories or grouping of management functions or tasks required, and will identify some of the key products that support the tasks. The degree to which you implement each function will vary based on your organization and systems management objectives; this is just to show you the tasks that should be considered when looking at Systems Management in the OS/390 environment.

30.2 Change Management

30.2.1 Overview

The change management discipline was discussed above as the example showing the effects of the environmental changes and the opportunities to exploit systems management disciplines to derive benefits from those changes. For a migration it is probably the most active discipline initially, simply because of the differences between the VSE and OS/390 environments - you have to change things for them to still work. After the migration, change management is still important - it will be driven by other disciplines such as problems (making changes to address/eliminate problem situations), performance management (making changes to improve the performance of applications, or allowing a resource to provide better performance to applications), and configuration (making changes to implement or meet the requirements of new levels of hardware and software), to name a few.

30.2.2 Tasks

The change management process includes the following:

- Collection - recognizing and gathering the changes, so that the focus is on changes as a whole, and not just on individual changes.
- Assessment - evaluation and approval of a change from both a technical and business standpoint.
- Planning - creating a plan that defines the steps in the change installation process. This also requires information regarding the current levels of hardware (including microcode) and software for system and network components, which is best met with a common repository for configuration data.

- Scheduling - occurs during the planning process, and aids in identifying conflicts and impacts, and determines target dates for changes.
- Distributing - this depends on the type of change, for example rolling out new levels of software across several systems.
- Installing - the actual installation of changes. Installation should be able to be scheduled for a particular date and time of day.
- Backout - reversing a change if it does not meet the installation or test criteria. The process to do this should be understood and planned for before the change is implemented.
- Tracking - transmitting the state of the change to the change management system. This provides a common view to everyone involved in the process.
- Post-installation analysis - reviewing completed changes to ensure they met the desired objectives, and to identify improvements that could be made to the change process.

30.2.3 Methodology

Change management is effectively implemented in several simple steps. First, a log of system changes should be kept. The simplest method is simply recording the information in a data set or PDS member. This can be improved by using a table (as provided by TSO/ISPF) or database management system (such as DB2) where change information can be formatted into fields for easier querying, searching, and updating. Ultimately one should investigate a change management product, which will also add the ability to support the more formal process activities, such as assessment, planning, and tracking, in an efficient, automated fashion. IBM′s TME 10 Information Management includes such functions, and allows change information to be updated and reviewed from a wide choices of platforms besides TSO - even via an automation product such as TME 10 NetView for OS/390.

One clear recommendation is that every system change, from system hardware to application software, be recorded and retained for a reasonable period of time -- at least a year. This will allow analysis of changes and discovery of any trends that are occurring (for example, are their certain changes that always lead to problems? If so, perhaps a better way to introduce those changes should be investigated). In addition, peer review of planned changes to the system should include review of the change control entries that document the change(s).

30.3 Problem Management

30.3.1 Overview

The Problem Management discipline is very closely related to Change Management. When problems occur, either a change may be needed to fix the problem and keep it from re-occurring, or a change was what caused the problem in the first place. Managing problems is critical to achieve high levels of application availability (which depends upon system and component availability).

Problem resolution is often assisted because of the existence of change management and problem management databases, showing previous instances of the same or similar problems in one case, and changes that took place in the system just before the problem occurred.

30.3.2 Tasks

The problem management process includes the following:

- Problem determination - the detection of the loss or impending loss of availability of a system resource to its users, and the isolation of the detected problem to the failing hardware, software, or microcode component.
- Problem diagnosis - the determination of the specific cause of a problem and the action required to resolve it. Diagnostic data gathered during problem determination provides input to this step. It may be necessary to gather and analyze additional information to complete problem diagnosis.
- Problem reporting - logging or calling the appropriate group (for example, a help desk) to have a problem logged for follow-up and solution.
- Problem bypass and recovery - the bypass of a failure, if necessary, until a problem can be resolved. The decision to bypass a failure is determined by the criticality of the lost resource and the cost of providing the bypass. If continuous operation is a requirement, recovery from a problem must take place immediately following problem determination and diagnosis. Bypass and recovery procedures should be automated whenever possible.
- Problem assignment - directing the problem to the proper resolver, as defined by enterprise policy.
- Problem resolution - the action taken to correct a problem. Once a problem is resolved, any steps taken to bypass it may be undone and the original resources placed back in service
- Problem tracking and control - tracking of problems from detection until their final resolution. Many symptoms may result from the same problem, and different problems may be related. Problem tracking allows the correlation of related symptoms and problems and helps to ensure timely recovery. Escalation of problems that exceed the established policies is a critical part of this step.
- Problem closing - specific notation that the problem has been solved to the satisfaction of the reporter. Problem cause and type must be noted for management analysis.
- Problem analysis - analyzing problem trends to reduce the number and impact of problems is a required management activity.

30.3.3 Methodology

As with change management, problem management depends upon the keeping of records about previous activities - in this case, problems that have been reported or discovered. The problems can be reported by end users who find anomalous behavior, system operations personnel, application programmers, or systems programmers. Automation can also be used to report problems - for example, if a critical job abends, NetView can detect this and create a problem record in TME 10 Information Management, and can also monitor and update the problem record as other detectable events occur, or at defined time intervals.

It is important that all problems be recorded in a file or database. This will serve as a repository for all reported problems, their current status, and their ultimate resolution, whether resolved by application change, system software change, vendor provided maintenance, or other. As the problems are worked toward resolution, the problem management database must be kept up-to-date. Many smaller OS/390 installations simply use TSO/ISPF′s editor to create,

update and manage problem management records, but use of a searchable database technology, such as DB2 or a custom problem management package such as TME 10 Information Management will be very helpful.

30.4 Performance Management

30.4.1 Overview

Performance management addresses the effectiveness with which information system components work together in order to achieve the optimum throughput and responsiveness with a given hardware/software configuration.

All too often, systems are designed, implemented and tested without careful consideration of performance factors. Fixing performance problems after the fact is much more difficult and costly than considering these factors during the design. In critical applications which are forecast to be heavily used, modeling the performance characteristics before the design is finalized can be very profitable. It is necessary to record performance data on a regular basis. The OS/390 system provides Resource Measurement Facility (RMF) which will record system resource (CPU, DASD or tape I/O, and so on) utilization for every job and job step in your system, or for any defined subset of it. In addition to the RMF data, it is valuable to have references to business volumes (orders, total number of order line items, number of paychecks processed) which can serve as an independent variable for estimating future requirements.

Performance management includes both real-time performance monitoring and long term capacity planning and reporting. Capacity planning relies on data from real-time performance monitoring, and uses it to determine trends that will influence future resource and application performance planning. Specifically, the long term growth in system resource consumption for various classes of system resources is monitored and future requirements are projected and used to identify usability end-points for system components. CPU, DASD, tape, printer, and similar subsystems can be studied and system capacity needs can be managed on a scientific and business management basis.

Reporting also includes comparing performance and availability achieved against agreed to service levels. Working with the problem management process can identify the reasons reported attainment did not meet service levels.

30.4.2 Tasks

The performance management process includes:

- Capacity planning
  - ⇒ Defining and managing the availability of system resources required to meet anticipated service demand.
  - ⇒ Modelling systems to determine and validate their ability to provide needed service.
  - ⇒ Validating user requirements against trends in current service levels.
  - ⇒ Collecting workload requirements and merging them into service requirements for all resources, such as hosts, network, servers.
- Performance policy definition

⇒ Establishing performance specifications and policies.

- Performance execution and measurement
  - ⇒ Response time monitoring - what is the response time as seen by the user?
  - ⇒ Availability monitoring - what's broken?
  - ⇒ Utilization monitoring - who are the biggest users? What is the service times and queue lengths for key components and resources?
  - ⇒ Component delay monitoring - what are the bottlenecks?
  - ⇒ Performance tuning - how can the resources be used effectively?
  - ⇒ Tracking and control - what are the short term trends? alerts? long term trends? how can I tailor the volumes and data flow to the needs and resources of the system?

30.4.3 Methodology

Performance information must be available to view in real-time to get a snapshot of system status, and to quickly address any problem or potential problem situations. It must also be archived so that long term analysis can be carried out. It is best to save this information to DASD or tape - printing and saving stacks of performance information is a waste of time. It takes up too much space, you will never find the information you need in a timely manner, and the information you probably need at a given point of time will likely be in a report that has just been discarded.

OS/390's RMF subsystem will record system resource utilization information. OS/390 SMF (Systems Management Facility) records contain RMF data and data from other sources (JES, VTAM, NetView, NetView Performance Monitor) that contains performance information. Many IBM and non-IBM performance products that monitor specific resources can place their data into SMF records.

While users, I/O control clerks, operators or specialized jobs can capture the business volume information, the use of performance and automation products can help make this task less labor intensive. Understanding the relationship between user transactions or batch jobs and business units of work will allow business volumes to be directly derived from performance monitors, and automation can be used to collect and format the data as needed.

Tracking performance data must also include online transaction response time, DASD I/O service times, batch program elapsed times, and batch window start and completion times. NetView Performance Monitor can measure and report online transaction response time. RMF and SMF can provide data on DASD I/O and batch performance. TME 10 Operations Planning and Control (TME 10 OPC) schedules batch workloads and can provide data on batch program and batch window elapsed times.

Using a PC and a spreadsheet application to download and capture and report in tables and graphs the critical performance variables over time is a very practical way to start; junior systems programming personnel, senior operations personnel, or help desk personnel can keep this information. A more robust solution will be a performance reporting product such as IBM's TME 10 Performance Reporter; it can directly read many of the sources that contain performance data, such as SMF, and produce short term and long term analysis reports for use in capacity planning.

Projected trends recorded in this way represent accurate growth measurements. These projections can be used to identify needed changes in system configuration with sufficient lead time to permit orderly procurement and installation of new resources. This will provide the capacity required over time as your system grows.

30.5 Operations Management

30.5.1 Overview

Operations management includes tasks for planning distributing, evaluating, and controlling workloads. It also addresses the resource availability needed to support the workloads.

The daily tasks required to activate system components, start workloads, and monitor activities to discover any discrepancies are the heart of supporting the system users. In a small environment these tasks may be carried out by a small number of people; in many cases the systems programming and daily operations functions are carried out by the same people. In the OS/390 environment, the opportunity for greater growth leads to the separation of operational tasks from systems programming tasks. Operations becomes the ″first line of defense″ when a deviation from the norm occurs.

The operations management discipline is oriented towards defined workloads. It covers the activities of starting and stopping systems and resources (including, but not limited to, host systems, workstations, networks and databases) and receiving and responding to operational notifications. However, it encompasses more than the traditional operator′s console commands, messages, and responses. The objective is to allow management to set policies to manage workloads and resource availability, and to automate the interactions required to implement these policies.

Operations management should provide the flexibility to centralize control of some functions and distribute others. This ability, together with other enhanced operator functions, will reduce the cost of operations.

The requirements for operations management may include:

- Lights-out operation - automated operations, ranging from simple command lists to automate a trivial or repetitive operator function to applying Artificial Intelligence (AI) to automate operator decisions.
- Data protection - automated backup and archiving of data files.
- Monitoring - consistent, easy to use, graphical operations interface(s), that display system and network topology and resources.

30.5.2 Tasks

The operations process includes:

- Workload planning - definitions, analyses, and reports of the enterprise′s workloads, both actual and anticipated.
- Operations planning - determines the structure needed to support the availability of systems and resources for the defined workloads.

- Automated Operations - handles the complex operations job scheduling procedures to ensure that work is completed in a timely manner.
  - ⇒ Verifying that the resources needed for a scheduled workload are available; for example, that the required disk or tape volumes are available for a backup operation.
  - ⇒ Planning to ensure the availability of day-to-day operations items, such as printer paper, tapes, and control center equipment.
  - ⇒ Specifying operations policies and procedures, preventive maintenance schedules and procedures, and operations recovery procedures.
  - ⇒ Supporting output delivery as defined by service-level agreements.
- Workload control - distributes work-handling and work-processing responsibilities across systems. It includes monitoring, analyzing, and adjusting of work in those systems. Examples of these functions include:
  - ⇒ Translating workload policies into system specifics.
  - ⇒ Distributing workload policies to systems.
  - ⇒ Receiving work requests and distributing them to systems, based on needs and policies.
  - ⇒ Managing resources needed for a workload for example, ensuring a tape volume needed by a batch job is mounted and ready for use on a tape drive.
  - ⇒ Monitoring systems and resources to determine work progress.
  - ⇒ Responding to queries about the status and progress of work.
  - ⇒ Accepting, and responding to, notification requests for work-related events, such as job termination.
  - ⇒ Taking action on workload-related events, such as restarting or rerouting work.
  - ⇒ Managing the printing and delivery of hard copy output.
- Operations control - applying operations policies for exception conditions, resource shortages, and other situations.

30.5.3 Methodology

Operators must have documentation and tools to understand how the system and workloads are supposed to be set up and run, the instructions to carry out setup/execution tasks, information on what to monitor and look for, instructions on what to do when something goes wrong, and a callout list of who should be contacted for various situations. This information can be kept in a set of files or in a PDS for starters; while it is popular to have hard copy ″run″ books, these are much more vulnerable to becoming out of date. In either case, a strong maintenance process that is tied into the problem, change, and configuration process is required to ensure accurate and up-to-date information.

Automating operational tasks is one of the most productive activities to carry out in the OS/390 environment. Automation will reduce problem detection and bypass time, eliminate human error, and support higher availability by carrying out quicker recovery actions. Automation can also make operators more productive by carrying out more mundane and repetitive tasks, monitoring for situations, and doing initial recovery. One of the simplest tasks to automate is message suppression, so that only critical messages are displayed at consoles;

the MPF list can be customized to carry this out. For other automation tasks, IBM provides several products to support workload and operational planning and control:

- TME 10 NetView, while many think of it as a network management product, is also a robust automated operations product for detecting and reacting to messages and alerts (for a wide variety of platforms), and carrying out automated and monitoring actions. NetView can also provide an automated interface with products such as TME 10 Information Management and Performance Reporter for MVS, to integrate operations management data with data from other systems management disciplines.

  TME 10 NetView also provides a central point of monitoring and automation from/to other operating system platforms; for example, it can monitor and detect a problem in a LAN Server or UNIX workstation that is required to allow users to access OS/390 resident applications.

- System Automation for OS/390 (SA for OS/390) is a NetView application for automating scheduling, monitoring and recovery of various OS/390 workloads, ESCON I/O configurations, and local or remote OS/390 hardware platforms. For example, it can be used to IML and IPL another OS/390 system, start in proper sequence a set of CICS and DB2 regions, alter an ESCON director configuration, and provide a central workstation to monitor what is running across one or more OS/390 systems.

30.5.3.1 Automating Operational Procedures

You should look at automating as many operational procedures as you can. Automation will reduce human error and provide consistency to the procedure. All of the operational procedures identified in the preceding section can be automated to some degree by using things such as:

- MVS facilities, such as exits, the Message Processing Facility (MPF) list and Automated Restart Management.
- An automated operation product, such as TME 10 NetView, System Automation for OS/390, or a third party automation product.
- Specialized products with automation interfaces, such as TME 10 OPC/ESA, DFSMS, and TME 10 Information Management.

Automation should be planned and added in stages as your OS/390 environment evolves. Both systems programming and operations personnel should be involved. Procedures should be tested thoroughly before being put into production. General types of activities you should consider automating include:

- Console operations, such as suppressing messages and replying to WTORs.
- Workload scheduling, such as starting up and shutting down online subsystems and batch jobstreams.
- Event detection, to quickly discover problems or potential problems that will affect system availability (for example, the JES2 spool filling up.)
- Monitoring performance and availability.
- Distributing software - both system software and application output, such as reports.
- Data backup and restore for both system and application datasets.

The following books contain planning information for automation and illustrate sample automated operational scenarios using IBM Systems Management products:

- *TME 10 NetView Automation Guide*, SC31-8225
- *Integrated Centralized Automation/Advanced Operation*, GG24-2599

Using the products below to support operational tasks will allow you to support growing workloads, as well as growing numbers of OS/390 images, in an efficient manner.

- TME 10 Operations Planning and Control (TME 10 OPC) automates the scheduling and (if required) rerun of batch workloads. TME 10 OPC allows you to define your batch scheduling requirements and will develop a schedule of when batch applications will run, and where. It will also monitor the real-time environment and notify you of deviations from the schedule. It can schedule work across multiple OS/390 images, as well as other platforms such as AIX, UNIX, Windows NT, and OS/400.
- Data Facility Systems Managed Storage/MVS (DFSMS) allows the definition of data classes and associating data sets with those classes, and will carry automate data set allocation, migration, and backup actions specified in the class definitions.
- Adstar Distributed Storage Manager (ADSM) allows using OS/390 as a repository to contain backup and archived data from other operating system platforms, and to carry out the required backup/archiving actions without manual intervention.

30.6 Security Management

30.6.1 Overview

Who the valid users of a system are, and what resources they are permitted to use, are the foundation of security management.

Implementation of an overall system security philosophy requires identification of system users and the resources they have access to, and what should happen (if anything) when an unauthorized user attempts access to a secured resource. If this design is done as the system is being installed and implemented, it is a series of small increments rather than a major undertaking.

Remember that even in a highly secure system, some users must be trusted. The security administration processes should be established with adequate controls and backups.

30.6.2 Tasks

Security management involves the application of security policies through functions such as:

- Policy definition - creation, deletion, and control of security services and mechanisms.
- Monitoring - distribution of security relevant information.
- Event or exception reporting - reporting security-relevant events.

30.6.3 Methodology

In VSE, users with security needs frequently use one or another vendor security package, as IBM provides only simple access control and logging security. In the OS/390 environment, in addition to vendor program offerings, IBM provides the OS/390 Security Server (a follow on to the highly regarded RACF product).

The Security Server provides system security services to ensure secure access from batch and online user programs to flat files, VSAM files, and databases. Printout, job submission and other system facilities such as program source and load modules, TSO/ISPF functions, CICS Transaction Server, IMS, DB2 and other OS/390 subsystems all integrate with the OS/390 Security Server protection for their resources. A System Authorization (SAF) API interface is provided so that user application programs can use the Security Server to protect application-specific resources.

30.7 Configuration Management

30.7.1 Overview

Configuration management is concerned with the generation and maintenance of a configuration database that contains information of all physical and logical resources and their relationships. Configuration is not concerned with implementing or managing changes to the information system resources, but rather with data on the location of components (current topology), their identifying attributes, their status (for example, active, online), future planning, and the process for gathering the configuration data.

For OS/390 accurate knowledge of the hardware physical configuration (connections of systems to I/O devices, systems to other systems), logical configuration (system and subsystem names, cross system connection definitions), and software configurations (product releases, libraries, and where they execute) becomes increasingly important as the environment grows. Inaccurate or obsolete configuration data can impact system availability and waste manual time hunting down and finding the proper information. This data is used within other disciplines, so the tasks for managing and maintaining it are important to carry out.

Configuration management data requirements include the following:

- Standard data usage - a single definition of configuration data for each type of resource.
- Shared, common data - ability to share configuration data among people, application, and subsystems. Sharing with the asset management discipline is particularly important.
- Reliable data - the ability to dynamically update the configuration database.

30.7.2 Tasks

The configuration management process includes the following:

- Configuration design - designing logical or physical, hardware, software, and applications configurations.
- Environmental planning - determination of the physical specifications required to support a configuration.

- Configuration creation - building and maintaining a configuration description that is resource-specific.
- Updating configuration information - providing vital product data, topology data, and other configuration data when a change is made to the configuration.
- Accessing configuration information - providing information on the actual or planned resources, specific resource attributes, paths to or from a target resource, version information for software, and similar data.

30.7.3 Methodology

Configuring the OS/390 hardware, logical connection, and subsystem environment is straightforward and documented in this book. For example, products such as HCD and System Automation for OS/390 (which includes the functions of ESCON manager) can be used to define the hardware configuration and access paths. The critical element is ensuring that related definitions for hardware and software components are kept together somewhere. For example, a DASD volume has a UCB address, I/O device address, and volume name. It is located in a particular hardware device. It is accessible through one or more control units (and associated ESCON ports). It may be dedicated to a particular workload or business organization. It is part of a pool used by a particular storage class. Managing the configuration means pulling this information together so that these relationships are known and understood, and so that systems management activities can be carried out properly using this information. The scope of a problem or change involving that DASD volume is much better known when one can see all the ″pieces″ of the system where the volume is defined in some fashion. This applies to all components in the environment.

Centralizing configuration information is best done using a set of files, which may contain both text tables and diagrams. Maintaining this information across these files is important so that the other disciplines have access to accurate information. In fact, if the ″attributes″ of a component - the different ways it can be identified or connected within the system - are documented, part of the change management process can verify that all of the attributes are properly accounted for in any changes that affect that component.

TME 10 Information Management provides support for combining configuration definitions for components within its repository; it can be customized to support any attributes for a component that the installation desires. This allows not only direct access to configuration information, but indirect access through change and problem activities; for example, if a problem occurs with a component, TME 10 Information Management can automatically look up the component′s configuration record and show what other aspects of the system may be impacted by the problem.

As the environment grows to multiple MVS images it becomes even more important to identify and maintain multiple configurations consistently. Where possible system definitions should be consistent; when they cannot be, a consistent naming convention should be used.

30.8 Asset Management

30.8.1 Overview

Asset management provides for managing the information technology inventory of resources, including both physical and intellectual assets. The components that you will use to support your workloads in the OS/390 environment all have business attributes - just like any asset in your company - that have to be accurately known and tracked.

30.8.2 Tasks

The required activities for asset management include:

- Administrative data - inserting into the central configuration database information such as the cost of components, their contracted maintenance expiration date, the eligible suppliers, and so on.
- Ownership assigned - controlling who is assigned as the owner of the assets, to ensure reliable and up-to-date data.
- Life cycle control - following a resource from identification as a requirement through purchase, installation, depreciation, and finally, disposal.
- Integration with the change process - this data is typically not captured automatically, and must be gained through administrative applications. Any means possible to ensure consistent data by interaction with other manual or automated processes is required.

30.8.3 Methodology

This process is not as critical during a migration, since the emphasis is on ensuring the components are defined and working properly together to support the migrated workload. In the long run the asset information can help identify some of the costs of running the installation, and when from a financial standpoint it makes sense to change or upgrade components. Asset management information can be kept in files, a database system, or a product like TME 10 Information Management; it should be kept where it can be easily related to the configuration data and activities mentioned earlier.

30.9 Accounting Management

30.9.1 Overview

Accounting management allows the managers of the enterprise information system to account accurately and efficiently for system and resource usage against the registered users of the systems. This is commonly known as ″chargeback″. If you were doing this in the VSE environment you will likely want to continue this, so knowing where OS/390 produces accounting data and how to get to it will be important. What is accounted for and charged back varies widely by company.

30.9.2 Tasks

Accounting management activities include the following:

- Measurement - collection of actual usage and service-level data.
- Cost allocation - creation of billing and charge-back transactions, including interfaces to other administrative applications or processes.
- Allocating and tracking project and other support costs.
- Creating and managing billing systems.

30.9.3 Methodology

The majority of OS/390 resource accounting information is produced in SMF records, either by OS/390 itself or other products (IBM and non-IBM) that will write data to SMF. Products such as TME 10 Performance Reporter or third party products can read SMF data and produce reports useful for billing resource usage.

30.10 Summary

OS/390 is best managed using a structured approach to Systems Management. This chapter highlighted some (not all) of the processes important to ensuring well managed environment. There are various process methodologies that will help identify all the tasks, and products available to support those processes. As your environment grows, automation will be required to reduce the increasingly complex management effort. Using a task approach, as opposed to a technology approach, helps centralize the process activities to minimize duplication and miscommunication, and form the base for this structured approach.

There are many IBM publications that can provide systems management process and product information, and implementation examples. The *IBM Networking and Systems Management Redbooks Collection*, SK2T-6022 is a softcopy collection of hundreds of publications that will provide useful information and guidance on managing the OS/390 environment.

Chapter 31. Diagnosing System Problems

31.1 Problem Determination Tools

Several tools are available under OS/390 to help the system programmer diagnose problems. The majority of these tools are intended for system problems rather than application problems and are often activated under the guidance of the IBM or ISV support center.

31.2 Dumps

There are many different types of dumps available in OS/390 for various situations. SYSUDUMP and SYSABEND dumps are for application debugging and are generally formatted and written to a JES spool data set. These dumps are similar in function to that of a VSE dump obtained via an OPTION DUMP statement. SYSMDUMPS normally contain similar information for application debugging but are stored on DASD and formatted with IPCS. SDUMPS (often called SVC dumps) are used mostly by OS/390 and various subsystems and are written to a system dump data set and formatted by IPCS. Stand-alone dumps are taken when OS/390 no longer responds to commands from the operator console and has most likely entered a wait state or is in a loop. The system activity display (SAD) may be useful to help determine if the system is in a loop or wait state. A stand-alone dump is normally written to tape (or sometimes to disk) and later formatted with IPCS once OS/390 has been re-IPLed.

31.3 IPCS

The interactive Problem Control System (IPCS) is a tool provided in the OS/390 system to aid in diagnosing software failures. IPCS provides formatting and analysis support for dumps and traces produced by OS/390, other program products, and applications that run on OS/390.

31.3.1 Analyzing Dumps

When you submit unformatted dump data sets to IPCS, it simulates dynamic address translation (DAT) and other storage management functions to recreate the system environment at the time of the dump. IPCS reads the unformatted dump data and translates it into a more readable format. IPCS can identify the following:

- Jobs with error return codes
- Resource contention in the system
- Control block overlays.

IPCS also helps your own dump analysis. For example, you can:

- Format control blocks. IPCS inserts field names into the output and displays the data in columns by field.
- Browse unformatted dump storage. IPCS allows you to easily follow pointers to other locations in the dump. It also retains addresses of certain locations in the dump.

- Reduce the size of a stand-alone dump. You can reduce the size of a stand-alone dump as you transfer it from tape to a direct access storage device (DASD) for IPCS processing.

31.3.2 Traces

There are a number of traces available in the OS/390 environment:

- OS/390 System Trace. A low overhead trace that uses a System 390 architected instruction. The trace provides a summary of system processing and is normally running continuously. This trace is present in all dumps.
- Master Trace. Provides a log of the most recently issued console messages at the time of a dump. This trace is formatted by IPCS when analyzing a SDUMP or stand-alone dump.
- Component Trace. Shows processing within single OS/390 components and almost always used under the guidance of the IBM support center.
- Generalized Trace Facility (GTF). OS/390 provides GTF to record trace data to DASD, tape or in storage. Many OS/390 components and subsystems write trace records to GTF with the most common being VTAM. GTF needs to be started in order for the trace records to be recorded. In addition the system programmer can issue SLIP commands to write trace records to GTF. GTF trace records can be formatted by IPCS, in addition VTAM trace records that are written to GTF can be formatted by ACFTAP.

31.3.3 Analyzing Traces

Using IPCS you can format the entries of any trace in a dump or trace data set. You can also do the following with GTF and component trace records:

- Selectively format records without deleting the unformatted data from the buffer or dump.
- Find the system and time stamp for each record.
- Mix formatted GTF and component trace records without combining the unformatted data.
- Reduce the number of records in a trace data set.
- Extract trace buffers from dumps.
- Combine GTF or component trace records into a single data set from multiple trace data sets.

31.3.4 Using IPCS

IPCS is a TSO command and is normally invoked while a system programmer is logged on to TSO, however in some cases it may be preferable to execute IPCS as a job through a batch invocation of TSO. Since the IPCS command must be executed before ISPF is invoked, there would normally be a separate logon procedure for users of IPCS.

An IPCS dump directory needs to be allocated for each IPCS user. Each dump directory can manage multiple dump data sets.

31.4 JES2 Diagnosis

There are some JES2 mechanisms that can be used for problem determination.

- $TRACE: JES2 internal tracing can be activated via $S TRACE; $T TRACEDEF; $P TRACE commands. These are typically requested by IBM service personnel.
- SYMREC: Symptom records are recorded in the LOGREC file for some JES2 internally discovered problems. Use EREP to format these entries.
- HASP088 message: When JES2 terminated abnormally, a HASP088 message is generated. This multi-line WTO has registers at ABEND, a traceback of program linkage leading up to the error, and of course the reason for the termination.
- Analysis of maintenance level: The maintenance level of a given module as well as its memory address can be determined via $D MODULE(name). This typically will be requested by IBM service personnel.

31.5 SLIP

SLIP commands are a powerful tool that can be used to either take an SDUMP or write trace records to GTF. Some of the SLIP commands are PER (program event recording) such as instruction fetch and successful branch and may cause CPU degradation depending upon how they are used. SLIP provides additional functionality beyond what the VSE SDAIDS does. SLIP commands are generally entered by a system programmer at an operator console. If the SLIP command causes trace data to be written, then GTF needs to be started for the trace data to be recorded.

31.6 Performance Tools

RMF is provided with OS/390, in addition there are several vendor performance monitors.

RMF provides three levels of performance monitoring:

- MONITOR I - records data to SMF that is used for performance and capacity analysis. Data is formatted via an RMF batch program or by other products such as SLR or SAS.
- MONITOR II - runs under TSO and provides an interactive view of the OS/390 system (CPU, paging, storage, dispatching).
- MONITOR III - a powerful interactive tool that monitors the OS/390 system for bottlenecks in system throughput and provides a point and shoot view of overall and subsystem performance. Current interval and historical interval data is provided.

31.7 LOGREC

The LOGREC data set is functionally equivalent to the VSE recorder file. In addition to hardware errors OS/390 records software errors to this data set. This provides valuable information in the initial problem determination phase of diagnosing a problem. EREP is normally used to extract the required records. In

addition, IPCS can be used to format the in-storage LOGREC buffers while analyzing a dump.

31.8 SYSLOG

All system and job related messages along with all operator commands are written to the SYSLOG JES spool data set. SDSF (under TSO) can be used to view the SYSLOG to aid in problem determination.

31.9 DFSMS/MVS Diagnosis

DFSMS/MVS provides many tools to assist a system programmer diagnose problems. Each component provides its own diagnosis documentation.

31.9.1 DFSMSdfp

DFSMSdfp provides many diagnostic aids that can be used by a system programmer when diagnosing problems. The *DFSMS/MVS DFSMSdfp Diagnosis Reference*, LY27-9606 provides information on reading dumps, running GTF traces, SMS traces, error codes, and more for each DFSMSdfp component. The *DFSMS/MVS DFSMSrmm Diagnosis Guide*, SY27-9615 provides instructions for diagnosing errors such as building keyword strings to search for known component failures.

31.9.1.1 Analyzing Catalogs for Errors and Synchronization

Catalog entries might become unsynchronized, so that the information about the attributes and characteristics of a data set are different in the BCS, VVDS, and VTOC. These differences may make a data set inaccessible or otherwise unusable.

To analyze a catalog for synchronization errors, you can use the Access Methods Services (AMS) DIAGNOSE command. This command will analyze the content of catalog records in the BCS and VVDS, and compare VVDS information with the DSCB information in the VTOC. DIAGNOSE will also check for invalid data or invalid relationships between entries. For more information concerning the DIAGNOSE command, refer to *DFSMS/MVS Access Methods Services for ICF*, SC26-4906. For more information on backup and recovering catalogs, catalog diagnostic information, using DIAGNOSE output for analysis, refer to *Managing Catalogs*, SC26-4914.

31.9.1.2 Catalog Recovery

The following is from Flash 9741 ICF Catalog Recovery

There is a dependency on catalog availability for continued operation of online systems, batch processing, and time sharing. A catalog outage can be extremely disruptive.

There are several utilities available from IBM that can be used to back up and reload a catalog. The user could choose from:

- IDCAMS EXPORT/IMPORT
- DFSMSdss logical DUMP/RESTORE
- DFSMShsm BACKDS/RECOVER

No matter what utility is used to perform the backup and recovery of a catalog, the process isn't complete until the catalog is resynchronized with the data sets as they currently exist. Between the time the catalog was backed up and restored, data sets have been deleted and defined and the catalog has to be updated to reflect these changes. These changes to catalogs, or catalog events, are recorded in SMF records. The SMF record types for catalog events are:

- 61 - When records are added to the BCS during DEFINE
- 65 - When records are deleted or updated in the BCS
- 66 - When changes are made to the BCS during ALTER processing

With these records, the information for catalog re-synchonization is available, but there is still a need for a facility that can use this information to update the catalog quickly. Using these records without a tool designed to process them is possible, but is time consuming and laborious.

There are products available such as the *Integrated Catalog Facility Recovery Utility or ICFRU* - (5798-DXQ) that provide this capability. Combined with a regular program of catalog diagnostics and backups, proper recording, dumping and tracking of data from SMF, it is possible to shorten the time of a catalog outage.

31.9.1.3 Checking a VSAM KSDS for Structural Errors

AMS provides the EXAMINE command which can be used to analyze and report on the structural integrity of the index and data components of VSAM KSDS clusters and the basic catalog structure (BCS) of an ICF catalog.

- EXAMINE INDEXTEST examines the index component of a KSDS by cross-checking vertical and horizontal pointers contained within the index control intervals.
- EXAMINE DATATEST evaluates the data component of a KSDS by sequentially reading all data control intervals, including free space control intervals.

Messages describing errors or inconsistencies are generated during EXAMINE processing as that condition is detected. For more information concerning the EXAMINE command, refer to *DFSMS/MVS Access Methods Services for ICF*, SC26-4906.

31.9.2 DFSMShsm

DFSMShsm has its own set of diagnostic aids that can be used by a system programmer when diagnosing problems. The *DFSMS/MVS DFSMShsm Diagnosis Reference*, LY27-9608 provides information about DFSMShsm control blocks and data areas used during diagnostic and maintenance procedures. The *DFSMShsm Diagnosis Guide*, LY27-9607 provides instructions for diagnosing errors such as building keyword strings.

The *DFSMS/MVS Managing Data Availability*, SC26-4928 provides information about disaster prevention and the recovery of DFSMShsm data based on real experiences.

31.9.3 DFSMSrmm

The diagnosis document for DFSMSrmm is the *DFSMS/MVS DFSMSrmm Diagnosis Guide*, SY27-9615. It documents how to obtain diagnostic information, eliminate common sources of errors, using the DFSMShsm Problem Determination Aid (PDA) trace formatter program, and building keyword search strings.

Maintaining the DFSMSrmm control data sets and report creation is documented in the *DFSMSrmm Implementation and Customization Guide*, SC26-4932.

31.9.4 DFSMSdss

The diagnosis publication for DFSMSdss is the *DFSMS/MVS DFSMSdss Diagnosis Guide*, LY27-9609. It provides information for building keyword search strings, format of the DFSMSdss dump data set, and the DFSMSdss patch area.

31.10 Diagnostic Reference Publications

The following books provide more detailed diagnostic information, and are useful for diagnosing specific problems:

SY28-1082	*OS/390 MVS Diagnosis: Procedures*
SY28-1084	*OS/390 MVS Diagnosis: Reference*
SY28-1085	*OS/390 MVS Diagnosis: Tools and Service Aids*
SY27-9605	*DFSMS/MVS DFSMSdfp Diagnosis Guide*
LY27-9606	*DFSMS/MVS DFSMSdfp Diagnosis Reference*
LY27-9609	*DFSMS/MVS DFSMSdss Diagnosis Guide*
LY27-9607	*DFSMShsm Diagnosis Guide*
LY27-9608	*DFSMS/MVS DFSMShsm Diagnosis Reference*
SC28-1737	*OS/390 SMP/E Diagnosis Guide*
SY27-2639	*OS/390 Security Server (RACF) Diagnosis Guide*
SY28-1086	*OS/390 JES2 Diagnosis*
GC28-1790	*OS/390 JES2 Commands*
SC33-6592	*OS/390 RMF Diagnosis Guide*
LY43-0078	*VTAM Diagnosis*
LY43-0105	*TCP/IP for MVS: Diagnosis Guide*
G544-5462	*PSF/MVS: Diagnosis Guide and Reference*

Part 7. Converting your Applications

Chapter 32. Conversion Process

Converting a data processing installation from VSE/ESA to OS/390 is a complex process that affects all areas of an installation. Personnel must learn different procedures; operations work changes in many ways and applications that run under VSE require conversion before they run under OS/390. Even managing the migration project, which includes planning, allocating people and resources and tracking the migration process, is a complex job.

Migration includes the entire process of moving your installation from VSE to MVS. Conversion deals with the changes that an application running under VSE requires to enable it to run under MVS.

This chapter:

- provides high level information about the conversion process
- directs the reader to resources for additional high level information
- summarizes the information of the preceding chapters

Major migration tasks and where task information resides in this book:

1 Planning and installing the MVS system.

- Refer to Chapter 25, “Prepare the Migration Environment” on page 401.

2 Training personnel to work on the OS/390 system.

- Refer to 2.6, “Educational Requirements” on page 31.
- Refer to Chapter 27, “Orienting ICCF Users to TSO/ISPF” on page 437.
- Refer to Chapter 28, “Orientation to OS/390 Console Operation” on page 443.
- Refer to Chapter 29, “Orientation for Utilities” on page 455.
- Refer to Appendix A, “Education Information” on page 535.

Each chapter contains information on personnel, training or OS/390 system use.

3 Analyzing migration requirements and developing a migration plan that is specific for this site.

- Refer to 2.7, “Scope of Work and Challenges” on page 32.
- Refer to Chapter 3, “Developing the Plan” on page 41.
- Refer to Chapter 3, “Developing the Plan” on page 41 and Appendix A of the *MVS Migration System - Planning Guide*, SB11-8077.

4 Analyzing the VSE workload and developing a complete list of the applications to be converted.

- Refer to 2.7, “Scope of Work and Challenges” on page 32.
- Refer to 32.4, “Preparation Phases” on page 493.

5 Developing standards for application conversion that reflect your standards for the new OS/390 system.

- Refer to 3.3.7, “Standardized Conversion Deliverables and Automation” on page 51.
- Refer to 5.2, “Data Set Naming Considerations” on page 99.
- Refer to 25.4, “Set Up Standards, Procedures, and Documentation” on page 407.
- Refer to Appendix C, “DFSMS Naming Conventions” on page 543.

- Refer to *MVS MS - Production Standards*, LB11-8080.

6 Translating the programs, taking into account the differences between VSE and MVS for each programming language.

- Refer to Part 3, “Converting VSE Languages to OS/390 Languages” on page 247.
- Refer to the specific program, utility or database section in this book.
- Refer to 2.7, “Scope of Work and Challenges” on page 32.

7 Converting the job control language. Because VSE and OS/390 differ significantly in JCL structure and syntax, methods of data distribution and production methods, JCL conversion is normally the most complex task of any migration.

- Refer to Chapter 4, “Job Control Language (JCL) Differences and Considerations” on page 69.
- Refer to 2.7.3, “JCL Conversion” on page 33.

8 Transferring data files from VSE to MVS.

- Refer to Chapter 25, “Prepare the Migration Environment” on page 401 for information on system connectivity.
- Refer to 32.5, “Conversion Phases” on page 503.
- Refer 2.7.4, “File Migration” on page 35.

9 Testing converted applications under OS/390.

- Refer to Chapter 26, “Test Environments” on page 419.
- Refer to 32.5, “Conversion Phases” on page 503.
- Refer to Chapter 31, “Diagnosing System Problems” on page 473.

10 Handling the production workload under OS/390, ensuring the jobs are submitted smoothly while staff is still learning about MVS operations.

- Refer to Part 6, “Running Your OS/390 System” on page 435.
- Refer to 32.6, “Implementation Phases” on page 515.

For a migration process time line that shows the relationship between the various conversion tasks refer to 3.4.2, “Project Plan Example” on page 56.

32.1 Conversion Process Introduction

The following discussions follow the methodology used in the Cortex Migration System (Cortex MS). Migrations using this methodology are implemented through a phased project approach. The methodology has proven to be successful, is well documented, and provides for an orderly discussion of topics.

The conversion process of the migration project can be divided into the following major phases and phase groups:

1 Preparation Phases
- Project Management
- Application Inventory
- Conversion Specifications
- Tool Customization

2 Conversion Phases
- Initial Trial Conversion
- Regression Testing and Repeated Trial Conversions

3 Implementation Phases
- Actual Conversion and Switchover
- Initial OS/390 Operations

This chapter will address these phases along with the key tasks to be completed in those phases. This chapter has been sectioned as follows:

32.1.1 References

These materials provide sources of supplemental information for this chapter.

- *MVS Migration System - Planning Guide*, SB11-8077 describes the planning process for the MVS-MS. This guide is for the people who are responsible for planning and scheduling the migration and fitting the conversion that MVS-MS performs into the migration schedule. It is the basic book for the project manager and every technical person involved in planning and running both the migration and the conversion.
- *MVS Migration System - General Information*, GB11-8074 provides an overview of the IBM MVS Migration System and is for the people at an installation who will decide if MVS-MS will work for a particular environment. It describes both the advantages and limitations of MVS-MS, presents information on how MVS-MS works, and identifies some specific early planning concerns.
- *MVS Migration System - Planning Chart*, SB11-8090 displays the standard conversion tasks and subtasks relative to their duration and relationship to each other.
- *MVS MS - Production Standards*, LB11-8080, is for the migration team members most concerned with defining the target MVS environment. It presents general information on MVS and detailed information on the areas in which your installation must establish production standards.
- Chapter 34, “Customer Migration Example” on page 529 provides an overview of a customer migration. It also includes discussion on the use of a two phased approach to a migration project and migration services providers.
- *IBM White Paper Appendix C, “DFSMS Naming Conventions” on page 543* provides information on OS/390 data set naming conventions.
- *IBM Washington System Center Flash # 9741*, which can be accessed through IBMLINK, provides information on VSAM catalog limitations in OS/390 after Year 2000.
- 33.2, “Conversion Tools” on page 520 provides information about conversion service providers and conversion tools

32.1.2 Prerequisites

There are two key requirements that need to be satisfied before embarking on a migration:

1. The source code must be available for your applications. If the source code does not exist then it must be rebuilt.
2. A method to transfer the source code to the OS/390 system.

32.1.3 Recommendations

The following are recommendations that either apply to all phases of your migration or are not specific to any phase.

32.1.3.1 Project Management

In some cases it may make sense to hire contractors, temporary personnel or a service provider to perform tasks that will only be performed once and do not provide long term payback to the installation. These one time tasks may include project management, specific conversion activities and use of project specific tools. There are many tasks to consider during a migration. Careful consideration should be given to knowing the skills that are available to the project, the requirements for systems programming, other projects that are planned or in progress, and how augmenting these skills and personnel may or may not make sense.

32.1.3.2 Take Advantage of Conversion Tools and Automation

Executing a migration with a mass conversion tool and automated processes can reduce both the time and people required to migrate from VSE to OS/390. Where it is not a large task to convert three programs and two strings of JCL, it is a large and difficult task to increase the scope by one thousand and perform the same conversion.

The automation provided by the use of a mass conversion tool is unique. After an extensive period of analysis, which includes running both pilot conversions and dummy conversions, you can, in a final mass conversion, convert all of your VSE applications to MVS in a single automated process.

32.1.3.3 Manuals

The *MVS Migration System - Planning Guide* and the *MVS Migration System - General Information Manual* are key publications and should be among the first manuals ordered when planning or investigating a migration.

32.1.3.4 Secure OS/390 Skills

The key benefits of having experienced production or systems programmer skills are with the installation of and running the new system. It takes time to learn and become comfortable in the OS/390 environment. An additional benefit of having an experienced OS/390 systems programmer on site, whether permanent or temporary, is through determining what the new system will look like by defining standards and naming conventions.

32.1.3.5 Migrate the SNA Network Early

If the migration plan includes converting an SNA communications network, then consider migrating ownership of the network from VSE to OS/390 within the two months that precede operating system switchover. At this time, switchover minus two months, the OS/390 system should be positioned for and nearly production ready. Assuming there is connectivity between the systems, the testing phase path has been from VSE to OS/390. After the ownership change of the network the path is from OS/390 to VSE.

This is a good task to perform as early as possible before switchover. Switching ownership of the network early provides some important benefits, including:

- Reduces the size and complexity of the conversion tasks on switchover day through dividing the conversion into smaller subtasks.
- Eliminates having to fix network problems on switchover day.
- The operations staff get experience on the OS/390 console.
- If new products such as NetView are installed the staff also gets experience with these.
- Builds migration team confidence through the successful subtask conversion.

32.1.3.6 24x7 Installations

The major conversion challenge in a 24x7 installation is the limited window of allowable time for testing and switchover. Typically the biggest consumer of time during testing and switchover is the data or file migration. Therefore this is the area to focus on to achieve time reductions during conversion. Methods to shorten the window need to be found and exploited. In a 24x7 installation the impact comes from having to stop the VSE production operations during the window in which you are moving data. You can't copy data while VSE production is running.

One additional element that increases the duration of the data transfer, is that backups should be run in case a fallback to VSE is necessary. If you migrate using tapes, then these tapes can be used. The use of tapes can be quite lengthy. One way to shorten the window where tapes are used is to orchestrate the process right down to a drill to maximize the use of the tape drives, tapes and people.

Additional testing and switchover timing considerations are:

- Time must be allotted for recovery and backout in the case of problems.
- Changes to hardware configurations and JCL can increase the duration of conversion. Examples include RJE stations where you have to change the JCL in them and the configurations of the PCs that are doing file transfers.
- Screen scraper applications are also affected. End users will be accustomed to seeing PC screens that are in ICCF format. After the switchover the screens will be in CICS or TSO formats. These need to be reconfigured within the window for switchover.
- For large databases it may make more sense to copy disk to disk for backup and then convert the new volumes in place rather than use database utilities to move the data.

  One method to expedite the data migration has been the use of extra DASD to handle the bubble associated with copying the database for switchover.

32.1.3.7 Two Phase Approach

The migration project can be broken into a few logical pieces that may help its execution. One method that has been successful is to begin with a mini project, phase 1, to identify and resolve your inventory. Proceeding with a known inventory will allow more precise cost analysis (time, people resources and so on). The cost of a conversion is based on inventory. It also provides information about the effort that may be required to recreate source materials. There are tools and service providers that perform these services. The second phase is the actual implementation.

The Phase 1 output is also a standalone deliverable that can be very useful for Year 2000 preparation.

32.1.4 Assumptions

For the purposes of providing more specific guidance for conversion projects, an approach to the migration had to be determined. This is also true for the migration effort itself, an approach must be adopted. The topics discussed in the Conversion and Implementation Phases of this chapter required that a choice was made. In these discussions, we will describe the environment associated with using Mass Conversion methods and tools. More specifically, the Cortex Migration System (MS) methods and tools will be used.

32.2 Mass Conversion - Background, Benefits and Method

32.2.1 IBM MVS Migration System - Background

The IBM MVS Migration System (MVS-MS) was a conversion aid IBM licensed and sold in the mid 1980s through the mid 1990s that consisted of both a conversion method and a conversion tool.

IBM licensed this conversion aid from SISRO and sold it as the IBM MVS Migration System. When sold through SISRO the aid was, and is known as the Cortex Migration System (Cortex MS). Remote support (via telephone) for MVS MS and Cortex MS was and is performed by SISRO for the Americas and by SISRO SA for the rest of the world.

IBM stopped licensing the Cortex tool in the mid 1990s. Although there have been many changes to the MVS and VSE operating systems and improvements to the conversion tool, the methodology of planning and execution of the conversion has not changed significantly. Today, the Cortex MS tool remains available from Sisro Inc..

A collection of documents including the *MVS-MS Planning Guide* and the *MVS-MS General Information Manual* was produced by IBM to facilitate the planning and execution of migrating from VSE to MVS. These manuals are available through IBM.

32.2.2 Mass Conversion Overview / Benefits

Mass conversion is the major distinction of the CORTEX-MS process. It results in a single switchover of the entire VSE application portfolio to OS/390 over a weekend. Until the switchover weekend, all converted applications run in production under VSE. By the end of the switchover weekend, all converted applications run in production under OS/390. In the mass conversion, there is no overlap of VSE and OS/390 production.

Cortex MS simplifies migration by automating the conversion of programs and JCL, and the transfer of files from VSE to MVS. It recognizes the differences between VSE and MVS and builds an MVS version of the VSE material.

In the mass conversion, all applications items are converted together, one conversion step at a time. It is the opposite of a traditional progressive conversion, in which all conversion steps are applied to the same application item, one item at a time.

For example, in an item-by-item conversion, each program would go individually through the successive steps of inventory, conversion, compilation/link-edit and regression testing. Instead, in a mass conversion, the entire code inventory (all programs, sub-programs, macros, copybooks, and include books) will first be verified for consistency and completeness. Then it will be converted together in one step, compiled/link-edited together in another step, and finally regression tested.

After each mass conversion step, results are reviewed and validated not just one at a time, but hundreds or thousands of application items at a time. Result validation too, is performed in mass, using summary statistics to classify all messages by occurrence and by severity. Individual manual verifications are conducted on a sample of items that have the same message, to identify by sampling the cause of the message and decide on a global resolution.

The first mass conversion is a pilot conversion. It is used for analysis, rather than for obtaining MVS material. The following mass conversions will be trial mass conversions, which will deliver MVS test material with an increasing quality, as the project and CORTEX-MS custom modification progress. Batch and online may be converted together or separately, as both will progress at a different pace. The final and actual mass conversion will be started after MVS tests have been successfully completed. It will deliver the actual MVS production material. The actual JCL conversion may be scheduled one or two weeks before the actual program conversion, in order to apply final manual JCL modifications. There will also be a special one-time translation of all application development source code, but without any compilation or JCL generation. Every three to four weeks, the mass conversion will start from a fresh copy of the entire conversion inventory, in order to take into account the last VSE maintenance modifications. Between two supplies, additional mass conversions may be executed from the same supply, in whole or in part, in order to take advantage of the latest custom modification improvements.

Key elements of the Cortex-MS mass conversion methodology are:

1. Automated Conversion
2. Repetitive Conversion
3. Mass Conversion (Switchover)

32.2.2.1 Automated Conversion

There are several ways to automate some or all of the conversion. The automation that Cortex-MS provides is unique in that it is a mass conversion. After an extensive period of analysis, which includes running both pilot conversions and dummy conversions, you can, in a final mass conversion, convert all of your VSE applications to MVS in a single automated process.

Mass conversion is more conducive to automation than progressive conversion, due to the fact that it addresses the same conversion requirement for all converted elements at a time, instead of addressing all conversion requirements for one converted element at a time as in the progressive conversion.

When converting one element at a time it may appear faster to do it manually than to invest in an automated solution. Until you see how many occurrences of the same conversion requirement must be addressed, you cannot properly assess the value of investing in an automated solution.

Because of the automated conversion procedures, all of the Cortex-MS conversion steps can be iterative; in other words, you can proceed by trial and error and then refine the Cortex-MS customization. A major step in implementing and customizing Cortex-MS is to automate the conversion procedures to support as many iterations as necessary before switching over to MVS. With this support, Cortex-MS enables a small conversion team to handle the conversion process in a relatively short time with minimum disruption to operations and development. This automation results in workload and time frame reductions. The methodology ensures consistent and reliable results, therefore reducing the scope of conversion tests.

A high degree of automation is required to convert and switch an entire VSE application portfolio to OS/390 over a short period of time, as done in the single-switchover-weekend mass conversion approach.

32.2.2.2 Repetitive Conversion

The repetitive conversion is an iterative method in which the conversion is improved by refining the automated conversion process and the associated software instead of the generated MVS material. After a trial mass conversion, the generated MVS material is function tested in MVS. In the event that the conversion reports or MVS tests indicate conversion errors, the CORTEX-MS software is custom modified to perform the conversion without errors, and a new trial mass conversion is run again starting from a fresh copy of VSE source material. This procedure is repeated until all conversion errors are eliminated. The actual and final conversion, and the switchover to MVS do not start until trial mass conversions are error free.

This approach allows assessing the progress made with the customization of the conversion tools and the automation of the conversion process. Carefully monitoring that progress is key to the project management, progress tracking and foremost to understanding when the automated conversion process is ready for the final mass conversion and weekend switchover. The repetitive conversion allows greatly reducing the risk inherent to a single mass conversion and weekend switchover.

By using a fresh copy of VSE source material for each mass conversion, all changes and modifications applied under VSE are automatically taken into account without having to follow up the VSE maintenance and to duplicate it in

OS/390. Therefore, it is not required to freeze or follow up development, enhancement, or maintenance of applications during the conversion.

32.2.2.3 Mass Conversion (Switchover)

The actual mass conversion, called the switchover, often takes place over a single weekend. Before switchover, the production workload runs under VSE. After switchover, the production workload runs under MVS.

The switchover is the key advantage of using Cortex-MS as a conversion tool. It eliminates the migration problems related to running some applications under VSE and some under MVS for an extended period of time. It eliminates the need to maintain both MVS and VSE versions of files. It allows maintenance and development work to be done on the VSE versions of programs right up until the actual mass conversion, and the program converted is the latest working VSE version.

32.2.2.4 Automation Limits

Some non-standard source code modifications must be performed manually, either because they are too complicated to automate, or because they are in very low occurrences. In both cases, automating the process would not be cost effective when compared to a manual modification. Manual modifications complement the automated conversion (before or after). They can be applied as VSE positioning, or manual OS/390 conversion.

Modifications applied to the VSE version of the source code, which is then tested and installed in production under VSE is known as ″VSE Positioning″. This technique is consistent with the repetitive conversion approach because it does not require freezing any source code. For example, VSE COBOL programs must be manually positioned to:

- Move STOP RUN statements outside INPUT (or OUTPUT) SORT PROCEDURE,
- Remove any access to I/O buffers before the file is opened or after the file is closed,
- Remove some undocumented features of COBOL Report Writer,
- Remove the use of the DISPLAY verb to imbed lines into a report written with the WRITE verb.

Modifications applied to the OS/390 version after automated conversion are known as ″OS/390 Freeze″. Here a procedure is defined to automatically identify and manually duplicate the VSE maintenance to the OS/390 ″frozen″ source code throughout the project. Manual OS/390 conversion is limited to exceptions because it is incompatible with the repetitive conversion approach.

After switchover, all production is done with the MVS versions of your applications. The programs that Cortex-MS translates become genuine MVS programs and are not emulated.

32.2.3 Mass Conversion Tools

Software tools suitable for mass conversion must automate most of the conversion of VSE JCL, programs and files to native OS/390 without any custom modification.

In addition, to accommodate the large diversity of coding patterns between VSE sites, the mass conversion tools must be flexible. Run-time options and exit

routines must allow users to custom adapt the tools to specific local conversion requirements. These requirements, not addressed by standard processing, are always identified. The conversion tools must be custom modified by positioning execution options, coding exit routines, and possibly developing some ad-hoc pre- or post-processors.

Finally, tools specifically designed for mass conversion must simplify and accelerate the review and verification of mass conversion results by producing summary statistical reports, which allow identifying and assessing at a glance the conversion of hundreds or thousands of items.

32.2.4 Automated Conversion Process

Assembling and customizing the mass conversion tools result in an automated conversion process, which converts the entire VSE application portfolio to OS/390 in only a few hours, including:

- Collection and verification of the application inventory
- Translation of application programs and associated macros, copybooks or subprograms
- Generation of OS/390 load modules
- Migration of the CICS maps
- Conversion of other production source items such as FCBs or PSBs and DBDs
- Conversion of VSE and POWER JCL streams including application and utility steps and associated SYSIN cards
- Generation of new OS/390 JCL complying with the selected OS/390 production standards
- Migration of VSE production files to OS/390

32.2.5 CORTEX MS

CORTEX Migration System (CORTEX MS distributed by Sisro Inc.) is the primary tool for mass conversions from VSE to OS/390. It consists of seven software components for converting VSE systems to OS/390. All of the CORTEX MS components are menu-driven through TSO/ISPF panels. CORTEX MS is installed either on the future OS/390 production system or on a temporary OS/390 conversion system. All seven CORTEX MS components are necessary to automate the mass conversion. Listed below is a description of the CORTEX MS software components:

DMT (DOS/OS/390 Translator)
: A translator for VSE JCL and programs written in COBOL, Assembler, PL/1, and RPG II.

INT (File Integration)
: Consolidates the results of JCL translation, classifies files according to their life cycle, and loads the Production Database (PDB).

PDB (Production Database)
: A batch application change control system that automatically generates OS/390 JCL to custom defined OS/390 standards. PDB uses an internal Production Control Language (PCL) for

functional descriptions of batch applications (standards and system independent).

EZ-PCL (Easy PCL)
A PC/MS-Windows based graphic user interface (GUI) to the PDB, which enables definition or modification of batch applications in their flowchart representation.

PREP (Preparation)
An interactive system of preparation, submission, backout, and restart of OS/390 jobs.

SWITCH (Switchover)
Transfers VSE files to OS/390 and DFSMS using VSE and OS/390 file information stored in the PDB.

ENV (Environment)
A set of subroutines supplied in source format that may be required to simulate VSE functions without an OS/390 equivalent: ISAM, COMREG, UPSI, and so on.

Some of the CORTEX MS/ENV simulation subroutines may be used for execution of the converted applications. Otherwise, no other CORTEX MS components are required for OS/390 operations.

Figure 57 illustrates the automated conversion process that can be developed, using the CORTEX Migration System, to convert VSE applications to OS/390.

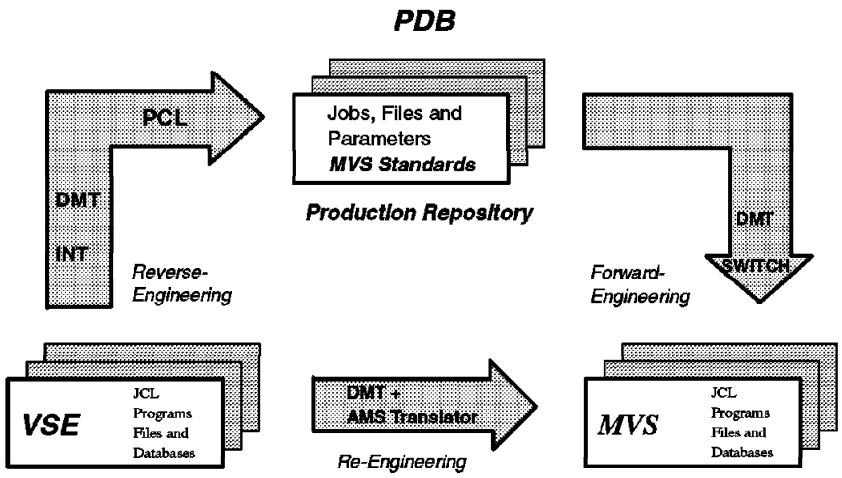


Figure 57. Automated Conversion Process

The discussions that follow address the functions provided by mass migration tools. The tool and process being referred to is the CORTEX MS aid developed by SISRO and licensed to IBM as the MVS-MS (Migration System).

The key functions provided by the mass conversion tool are:

- Inventory Validation
- Translate the Languages/Programs
- Translate the JCL
- File Transfer

32.2.5.1 Inventory Validation

The Cortex tool cross references the run JCL, the procs, the FCBs and various includes. Then it cross references the source programs, copy books, macros, subroutines, PSBs, the CICS tables and maps for CICS; basically all the JCL elements. In CICS the CICS PPT and transaction programs and applications are cross referenced. The tool cross references these against the base Processing Program Table (PPT). From these each element is linked to what the element includes to determine what is referenced, what is missing and which is unreferenced. The benefit of the tool is that when you run the Cortex scan utility the results are available in a few minutes versus a few days if cross referenced manually.

32.2.5.2 Translate the Languages/Programs

The Cortex MS tool can translate most syntax that is acceptable to VSE compilers, and most of the clauses that VSE and MVS compilers interpret differently.

COBOL Conversion Tools

The main support the Cortex MS tool provides regarding COBOL conversion is with Reserved Words.

Assembler Conversion Tools

The main task the Cortex MS tool provides regarding Assembler conversion is with Macros.

32.2.5.3 JCL Conversion Tools

Cortex MS converts most VSE JCL and job entry control language (JECL) statements and operands, and their different formats.

The main tasks the Cortex MS tool provides regarding JCL conversion are:

- Expand the JCL and convert it in association with the file information that comes from the programs
- Provide guidance on file handling
- Provide the capability for exception handling

32.2.5.4 File Transfer

The Cortex MS tool automatically handles the transfer of most files from VSE to MVS by building the REPRO jobs to copy the correct files to OS/390. The tool builds a listing, to be used for data migration, from reading VTOCs and LISTCATs. It also reads JCL, the programs and the source code from which to build cross references.

The tool does not provide any utility to transfer the data, only the utilities to prepare jobs to use IDCAMS. The IDCAMS REPRO standard utilities that exist in both VSE and OS/390 can provide this function.

32.3 Mass Conversion Phase Overview

Built on the principles of mass, automated, and repetitive conversion, a mass conversion project is organized in phases, as shown on Figure 58. The figure shows the specific phases and includes an approximate schedule with durations.

	Duration	From	To
Preparation Phases			
Application Inventory	2 m	*mid-JAN*	*mid-MAR*
Specifications	3 m	*start-FEB*	*end-APR*
Customization	3 m	*mid-FEB*	*mid-MAY*
Conversion Phases			
Trial Conversion (2 w to 2 d)	5 m	*end-APR*	*end-AUG*
Regression Testing			
Online	4 m	*end-APR*	*end-AUG*
Batch	4 m	*mid-MAY*	*mid-SEP*
Simulated Production	1 m	*mid-AUG*	*end-SEP*
Implementation Phases			
Actual Conversion & Switchover	1m + 2d	*end-AUG*	*end-SEP*
Initial OS/390 Operations	1 m	*end-SEP*	*end-OCT*

Figure 58. Project Phases

32.4 Preparation Phases

The project phases included in preparation activities are:

- Project Management
- Application Inventory
- Conversion Specifications
- Tool Customization or Tailoring

The main activities of the preparation phases includes:

- Analyzing the VSE workload to be converted
- Developing the migration workbook
- Planning the conversion implementation

References you can consult for additional information about the preparation phase include:

- Refer to Chapter 3, “Developing the Plan” on page 41 for information on project staffing, assignments and responsibilities.
- Refer to Appendix C, “DFSMS Naming Conventions” on page 543 for information on data set naming conventions.
- Refer to Appendix A of the *MVS-MS Planning Guide* for help developing the Migration Plan.
- Refer to Appendix B of the *MVS-MS Planning Guide* for help developing the Conversion Plan.
- ***Implement System Managed Storage (DFSMS)***

It is strongly recommended that DFSMS be planned for and implemented from day one of the migration. The key benefit of implementing DFSMS is how it helps the installation be positioned for future growth. Implementing DFSMS also helps the conversion by providing structure for standards and naming conventions. Other benefits of installing DFSMS from the outset are:

1. No future conversion of the configuration is required.
2. Positioning for new software function and hardware support where most new functions in OS/390 have SMS as a prerequisite. Examples include data compression and extended data sets.
3. Provides DASD management.
4. Supports cataloging all your data sets from the beginning.
5. Less complicated JCL from a VSE user perspective through the use of data classes.

32.4.1 Phase 0: Project Management and Technical Leadership

A Project Manager/Technical Leader is assigned the responsibility for administration and technical direction of the conversion efforts. Project management activities consist of:

- Developing detailed work plans and schedules
- Evaluating the progress made against work plans and schedules
- Conducting weekly progress review meetings
- Conducting monthly project status reports
- Providing liaison and coordination between all personnel involved

32.4.1.1 Project Planning and Orientation

The objective of this task is to develop detailed project plans and orient the conversion team and all personnel involved to the conversion approach and to the project plans. The project plans include:

- Overall project plan
- Online application test plan
- Batch application test plan
- Switchover plan

It is recommended to use a project management software tool to develop and update the project plans. An example of a project plan is provided in Chapter 3, “Developing the Plan” on page 41.

The project education sessions are scheduled:

- At project start
- Before the start of online application tests
- Before the start of batch application tests
- Before switchover

During those sessions, the Project Manager and perhaps, hired conversion specialists, provide the conversion team with instructions and guidelines for planning, organization, and implementation of the activities to come.

32.4.2 Phase 1: Application Inventory

Before you start any work on a migration you need an inventory. Start the inventory process as soon as you have made the decision to pursue migration. The work that results in a clean inventory is about determining what is in production and what is not in production on your system.

Taking inventory of your applications is a basic migration task. It does not require tremendous skill but can prove to be very laborious to complete. It is a necessary prerequisite for estimating the costs of the conversion tasks.

The objective of this repetitive task is to identify and collect the conversion inventory, transfer it to the OS/390 system, and verify that it is complete and consistent. The conversion inventory includes:

- Source code: programs, subprograms, macros, copy or include books, and so on
- JCL: VSE and POWER JCL streams, standard labels, SLI and other JCL include books
- Additional information such as job scheduling, CICS tables, VSE catalogs and VTOC listings

As a first step for the VSE to OS/390 conversion, the application inventory must be collected and verified for completeness and consistency. This allows the conversion process to begin with clean libraries, resulting in a smooth and efficient project. An iterative process, this validation is completed in two to three months. Up to four iterations of the following application inventory tasks are typically required:

- Developing/refining an inventory transfer procedure
- Loading the conversion inventory into the conversion libraries
- Executing the inventory validation software
- Identifying any missing or unreferenced elements in the conversion inventory
- Resolving the missing and unreferenced elements in the conversion inventory

It is recommended that job schedulers be taken advantage of as they can be very helpful in keeping track of what is current on the system. From the scheduler, lists of production jobs can be extracted. They can provide a good starting place from which to begin inventory validation. In CICS for example it is common to have CICS tables that are full of obsolete material.

The Application Inventory phase is complete when the application inventory contains only a small percentage of missing or unreferenced elements.

The key terms associated with determining your application inventory are:

1. Determination
2. Collection
3. Supply
4. Analysis and resolution of exceptions

32.4.2.1 Determination

Determination is the task of understanding what runs in production on the VSE side. It includes finding out all the places where the production JCL is stored and determining what is production and what is not production.

32.4.2.2 Collection

Collection is about building a procedure using standard utilities or ad hoc programs to transfer all the source (JCL, source programs and copy books) to the OS/390 side.

32.4.2.3 Supply

Supply is the procedure where you transfer the source code and JCL, from the source environment, VSE or VM, to the OS/390 system. The determination, collection and supply happen on the VSE side.

Only the version of source code or JCL currently used in production under VSE is supplied to the conversion. Duplicate or obsolete versions are eliminated (moved away) from the VSE production libraries. VSE executable code (phases) is discarded: new OS/390 executable code (load-modules) will be generated from the converted source code. Lost source code is either retrieved or rewritten: it is then regression tested and installed in production under VSE before being transferred to the OS/390 system for conversion.

The device, content, and format of the files used to transfer the conversion inventory from the VSE to the OS/390 system are defined. An automated application transfer procedure (VSE JCL streams and OS/390 JCL streams) is developed. The conversion inventory is collected from the VSE production libraries, copied to transfer files and downloaded into the conversion libraries on the OS/390 system.

In the mass method the supply is renewed each month. This helps synchronize both sides by keeping the VSE portion more current and ensuring the MVS side has access to a recent VSE copy.

32.4.2.4 Analysis and Resolution of Exceptions

This is also known as Inventory Validation. It involves analyzing the relationships in the inventory and uncovering and resolving exceptions. Validation is the process of determining to what each element is linked, what those elements include and to follow the chain to see what is referenced, what is missing and what is unreferenced. Missing elements can be identified where a piece of JCL calls for program ABC but the source ABC is missing or the piece of JCL is not production JCL. The same is true with unreferenced elements. When you have unreferenced elements there are two causes. It could be an obsolete item or the piece of JCL that referred to it is missing.

The end result of resolving these exceptions is a clean and accurate inventory.

In the Cortex environment this is arrived at through the running of validation tools against the conversion libraries on the OS/390 side. Validation could also

be performed on the VSE side and then sent to OS/390. The determination and collection procedures are developed once and then repeated. The supply is done once per month. Also unique to the Cortex MS environment is that the analysis and resolution work is on going throughout the project.

The Cortex tool produces reports listing missing and unreferenced elements to assist with resolving exception conditions. Those exceptions are reviewed and addressed by VSE production support personnel. There should be no missing elements in the supply of the conversion inventory by the middle of the specifications phase.

Sometimes grouping the exceptions by names provides a clue to their condition. The next step is to analyze why these conditions exist and then resolve the way the supply is produced by adding or deleting members or by changing the supply procedure to pick members or code from other places or edit it. This is an iterative process that is repeated every three weeks during the conversion. The schedule ensures that the supply comes clean and remains clean.

The most common mistake made during inventory validation is to casually delete unreferenced elements. Unreferenced often indicate that something has been missed upstream.

Another common problem is to have production jobs where the JCL resides in someone′s desk. These can provide ad-hoc jobs that even the scheduler was unaware of. This is also true of jobs that are submitted from remote locations through RJE.

The first task is to validate and correct the procedure for supplying the VSE source material and the design that is included in the conversion plan. A second important task is to validate the VSAM user catalog file definitions regarding VSE/MVS compatibility.

The key point here is that the quality of the final product of the validation is only as good as what is fed to it.

The conversion inventory is mass collected-transferred-verified-converted every two to three weeks from project start to switchover, in order to automatically take into account the VSE application changes. Because it will be repeated many times, this mass processing procedure must be automated to reduce or eliminate the manual effort and to ensure repeatable, reliable and improving quality from a capture to the next one (less or no missing elements).

The application inventory is collected for a final mass conversion a few weeks before switchover. Between this final mass conversion and the switchover, several captures are scheduled to identify (source compare) and carry-over the last VSE application changes. But the changed elements are converted (or their changes are duplicated) one at a time, to eliminate the risk of massive last minute regression with the automated conversion.

32.4.3 OS/390 Standards and Naming Conventions

The objective of this task is to define a set of OS/390 standards and naming conventions for the converted applications.

It is recommended that you define the new OS/390 standards and operating procedures first, at least at a high level, before defining new JCL-referenced

names. This is because naming conventions can facilitate or impair the implementation of OS/390 system components (such as DFSMS) and other automated operations tools. It is recommended that you understand how a production item will be stored, used and managed under OS/390 before giving it a name.

Because new OS/390 JCL streams will be generated, new rules are developed to define the structure of those streams, and how repetitive JCL statements will be regrouped into procedures or includes in order to facilitate future JCL maintenance. For example, a dedicated JCL procedure may be used for each of the highly repetitive utility steps (SORT, IDCAMS, and so on), and compiler or debug related DD statements (SYSOUT, SYSUDUMP, and so on) may be regrouped into a single JCL include. The storage and management of control cards and their reference from the JCL is another set of rules associated with the structure of JCL streams, as is the usage of execution parameters and their transfer from production control clerk to executing program through job scheduler and JCL symbolic parameters. It is recommended that you not redefine the division of batch applications into job streams, because this would impact the job scheduling rules and confuse operations personnel.

As part of the OS/390 migration, the usage of utility steps can be redefined. It is typical to find half a dozen ways of performing the same file copy in the VSE JCL. Only a couple of ways may be needed and retained in the OS/390 JCL. The same standardization applies to other file utility steps, such as external sorts and backups. Database application steps and database utility steps are also standardized, and some of the associated JCL may be isolated in JCL includes or procedures for easier maintenance. Service steps of all kinds may be inserted automatically by the mass conversion JCL generation tools, for example to delete catalogued temporary files.

System managed storage is very strongly recommended when it comes to file management under OS/390. One of the great combined advantages of DFSMS and DFP is that they allow drastically simplified DD statements, which makes JCL very easy to read or maintain, and be configuration independent. Typical OS/390 migration standards include the elimination of VOLSER, RECL, BLKSIZE, RECFM, ORG, DSCB, SPACE and UNIT attributes (with the exception of RECL in IDCAMS/REPRO output and UNIT for tape file output).

Many VSE sites tend to be tape oriented. The OS/390 migration is an excellent opportunity to migrate from tape intensive to disk intensive operations, eliminating many manual interventions (for tape volume mounts) and boosting job throughput as a result. DFSMS′s automated free space release and HSM′s archival features can be combined to implement disk space ″buffering″ techniques, in which files that will end up on tape are created on disk by the executing job. The copy of the file to tape, as well as the accumulation of many files on the same tape volume is left to HSM. It takes place after the job execution is completed. Once again, the effect is a drastic reduction of manual interventions for tape mounts, as well as an acceleration of the job throughput. Such file and device management policies are part of the standards and operations procedures definition in an OS/390 migration. They have an enormous effect on the structure and content of JCL streams.

New names must be defined under OS/390 for all items referenced in the new OS/390 JCL, including data sets, jobs, procedures, includes, steps, execution parameters, libraries of any kind (from source or executable code to procedures

and parameters), and library members for control cards. This is because many of the VSE names are syntactically incorrect under OS/390. But the conversion of in-house developed applications doesn′t require changing any of the names associated with the source code. In fact, it is recommended to leave the program, entry-point, copybook, code include and DD names unchanged to avoid confusing application support personnel.

The process of developing standards includes striking a balance between the conventions and familiarity of the VSE system and adopting durable and usable standards for the future OS/390 system. Follow local conventions wherever possible. Operators are used to particular names of data sets. Maintaining these wherever possible will make the transition easier.

32.4.4 Phase 2: Conversion Specifications

The project-specific conversion requirements are determined and documented during the specifications phase. The specifications phase, which begins with a ″clean″ supply of the conversion inventory (source code and JCL), is typically completed in two to four months. The specification tasks include:

- Developing a project plan
- Educating the conversion project team to the conversion approach and to the project plan
- Installing the conversion tools and performing a preliminary tailoring to local conditions
- Studying the VSE source production environment
- Defining the OS/390 target production environment, including OS/390 standards and naming conventions
- Performing a pilot conversion of a subset of the conversion inventory before custom modifying the conversion tools
- Identifying conversion issues and source code positioning activities
- Designing automated conversion solutions, based on custom modification of the conversion tools, or development of additional ad-hoc conversion software
- Conducting specifications meetings with technical representatives of VSE and OS/390 operations, technical support, and applications development

The site′s specific conversion requirements are determined and documented by analyzing the VSE source material, designing OS/390 target material that complies with the selected OS/390 production standards, and designing the conversion paths (program translation, JCL conversion, and file migration).

This process comprises knowing what software will be replaced and what standards will be used in OS/390. It is recommended that DFSMS based naming standards be used in the design of the MVS target system. These must comply with the installation′s standards and operations procedures for the MVS target production.

The Specifications phase is complete when the conversion specifications have been developed and approved by operations, technical support and applications development.

References you can consult for additional information about the conversion specification phases include:

- Refer to Appendix C, “DFSMS Naming Conventions” on page 543 for information on data set naming conventions that relate to an DFSMS environment.
- Refer to Appendix A of the *MVS-MS Planning Guide* for help developing the Migration Plan.
- Refer to Chapter 3 of the *MVS-MS Planning Guide* named ″Developing the Conversion Plan″.
- Refer to specific product, program or utility migration guides. Examples include COBOL or DB2 migration guides.
- Refer to *MVS-MS Production Standards* document for information on JCL standards. It does not provide guidance on programs. It does provide a model of how to structure your jobs, job names, job step names and data set names.

Recommendations for the specifications phase include taking a training class on the installation of System Managed Storage (SMS).

The major elements of the Specification phase are:

1. Analyze the VSE source material
2. Design the MVS target output
3. Determine the method to get from source to target

32.4.4.1 Analyze the VSE Source Material

This task is a process of looking at the source material and referring to the checklist of specification considerations. Your specification document can be based on the checklist in the *MVS MS Planning Guide - Appendix A* . From this comparison you can identify the problems or exceptions needing to be resolved by the conversion team. You may find you are lacking a key utility, or have not identified a plan to deal with a unique operation or that you have odd JCL.

The analysis of the VSE material is both qualitative and quantitative. Qualitative analysis consists of listing the types of syntax that are being used in VSE, and defining the types of replacement syntax that will be used in OS/390. Quantitative analysis consists of determining the number of times that each type of syntax occurs within the VSE material.

Quantitative analysis is performed with scan utilities. It is essential in determining the conversion approach to be used. While conversion issues that occur numerous times are addressed with automated solutions (if technically possible), low occurrence conversion issues may be addressed with manual positioning of the VSE material (when possible) or even manual modification of the OS/390 version.

The resolution can come during meetings with the services provider and/or site team where explanations can be presented of why certain things are done the way they are or why things are where they are.

32.4.4.2 Design the MVS Target Output

All the material in this book, including the charts that show functional comparisons of products, is for aiding the analysis of the VSE system to help determine the target OS/390 system. It is during conversion team meetings that these items are presented, challenged and designed.

For project organization and planning, the effort is normally divided per type of application item (JCL, code and files), per type of application (batch or online), and per language (COBOL, Assembler, RPG II, and so on). The design of the OS/390 target material complies with the selected standards and operations procedures for the OS/390 target production.

The conversion specifications are documented in a Conversion Specifications Document that becomes the guideline for custom-modifying the conversion tools and developing or custom-modifying additional conversion tools.

32.4.4.3 Determine the Method to Get from Source to Target

Determining the method to get from source to target is the outcome of these discussions with the conversion team. The outcome becomes the basis for your conversion plan.

There are also implications on OEM products that perform functions such as Report Manager, schedulers and backup utilities. Include these topics in your discussions.

32.4.5 Phase 3: Customization or Development of Conversion Tools

This section is specific to the conversion tool. Customizing the tool is unique to the mass migration method and is a cornerstone of the Cortex tool. In the Cortex method customization is how you deal with exceptions or what you do when you don't like the tool's output. In Cortex, you modify the tool and rerun your input.

The objective of this task is to adapt the conversion software or develop additional conversion tools to be able to automatically convert all or most of the VSE material to OS/390, and deliver OS/390 material that complies with the selected OS/390 standards and operations procedures.

The custom modification of the automated conversion process follows the specifications documented in the conversion specifications developed above. It is implemented through the positioning of conversion tools options, design and coding of exit routines, design and coding of ad-hoc pre- or post-processors which are then added to the automated conversion process.

Similar to the definition of specifications, this effort can be divided for better project planning and organization per type of application item, between batch and online and per language.

The customization is validated by converting representative samples of VSE programs and JCL and verifying that the local VSE syntax has been replaced by the appropriate OS/390 syntax.

The Custom Modification phase, which begins simultaneously with the second third of the specifications phase, is typically completed in two to four months. The Custom Modifications tasks include:

- tailoring and custom modifying the conversion tools through installation options and exit routines according to the conversion specifications
- performing a pilot conversion of a subset of the conversion inventory after having custom modified the conversion tools
- auditing the messages and OS/390 material (source code and JCL) produced by the pilot conversion
- performing technical tests on a representative sample

The Custom Modification phase is complete when the conversion process automatically handles the conversion requirements as defined during the Specifications phase.

Two additional items are associated with the objective of Custom Modifications. They are:

- Manually modifying some code for the areas where the tool will not be used
- Positioning the VSE source to remove incompatibilities

VSE Positioning

VSE Positioning is about modifying the VSE code to eliminate VSE to OS/390 conversion requirements that cannot be automated.

The positioning which doesn't impact the logic of the code, can be applied by hired consultants. The positioning which impacts the logic of the code (misplaced stop runs, access to file buffers before open or after close, and so on), is best left to application support personnel familiar with the code.

In any case, the positioned VSE code is then regression tested under VSE by application support personnel and rolled back into production under VSE. It is collected later on for conversion, together with the rest of the application inventory.

Manual OS/390 Conversion

The objective of manual conversion is to complete the automated conversion process by applying manual modifications to the OS/390 version of the code, when fully automating the conversion is too complex or is not cost effective, and when VSE positioning is also impossible.

Manual conversion activities may be required for the conversion of VSE-dependent Assembler programs, part of the COBOL to COBOL for OS/390 conversion, and a few low occurrence JCL conversion issues.

Manually converted elements are subjected to a one time automatic conversion to take into account all the conversion requirements that can be automated. Then they are set apart from the automated conversion process. They are repetitively supplied with the rest of the application inventory for mass conversion. Instead of being re-converted each time, they are only source compared (source compare utility program) to the version manually modified, in order to identify the latest VSE version changes, if any, and duplicate them manually in the OS/390 version.

Because such activity is in contradiction with the mass, automated and repetitive conversion approach, it is kept to a minimum number of application items that

cannot be converted entirely automatically, and for which the unresolved conversion requirement cannot be addressed through VSE positioning.

32.5 Conversion Phases

The specific phases included in the Conversion Activities are:

- Initial Trial Conversions
- OS/390 Regression Tests and Repeated Trial Conversions

The objective of the conversion phases is to automatically mass convert the inventory of production source code and JCL to OS/390, to deliver functionally equivalent OS/390 material (source code, load modules, run JCL, and JCL procedures), and to generate VSE to OS/390 file migration JCL procedures.

The phases end when the quality of the conversion is deemed satisfactory and all critical applications such as daily, weekly, and online applications have been successfully tested under OS/390.

32.5.1 Program Conversion

The mass conversion key steps relative to program conversion are:

Program Conversion Generates OS/390 source code and extract some information required for VSE JCL conversion, such as: files opened, their device types, assignments, block sizes, and open mode (input, output); entry-points; referenced macros, copybooks, includes or sub-programs.

Compilation/Link-Edit Starts from the OS/390 version of the source code to generate executable code (load modules).

32.5.1.1 Program Conversion Considerations

The following considerations are recommended to be performed during (or prior to) your migration to OS/390.

- After all VSE programs are identified and collected for the conversion process, they must be compiled in VSE to detect if there are any source/object discrepancies with VSE production. This ensures that the correct level of the program (source) is used.
- To be as compatible as possible to OS/390 requirements, VSE programs should be compiled using the latest levels of the VSE compilers. Migration to the Language Environment for VSE and the Language Environment enabled language compilers is strongly recommended.
- Any VSE program without source code must be rewritten. Products are currently available that will allow source to be recovered (or reconstructed) from executable modules.
- RPG CICS programs, if any exist, must be rewritten in another high-level language.
- All ISAM programs must be converted to VSAM.
- BDAM programs must be converted to relative record VSAM where possible.
- VSE programs using the UPSI and/or the DATE statement facility will have to have this function simulated in another fashion. In OS/390 either use the EXEC card ″PARM″ facility (similar to the VSE PARM parameter introduced in

VSE 2.1 for the EXEC statement), or change the program's logic to read and process a "control record" which would supply any variable information to the program.

Additional programming changes and considerations can be found in their respective programming chapters and in the POWER-JES2 differences chapter. The Cortex Migration System can provide conversion assistance in many of the areas where VSE-OS/390 incompatibilities exist.

32.5.1.2 Common VSE Coding Practices Causing Conversion Problems

The following are some common VSE program coding "practices" that won't work in OS/390, and since they are mostly logic errors, won't be "picked-up" nor notated by a conversion tool.

- In COBOL, referencing a file's (or printer's) I/O area before the file (or printer) is OPENed. This will result in a system 0C4 abend in OS/390.
- In COBOL, referencing a file's (or printer's) I/O area after the file (or printer) is CLOSEd. This will result in a system 0C4 abend in OS/390.
- In COBOL, "STOP RUN" statements should not be embedded within SORT procedures. These should be removed from all SORT procedures; that is, sorts must end before a STOP RUN can be requested.
- Not OPENing a unit record file will work in VSE, but abend in OS/390.
- In Assembler, using other than registers 2 through 12 for application purposes. (Registers 0, 1, 13, 14, and 15 are used for special purposes by OS/390.)

32.5.2 JCL Conversion

Mass conversions use the following steps for JCL conversion.

- Conversion

  The file information extracted during program conversion and associated data such as VSE disk and tape catalog data, is used by the conversion tool to generate individual job stream flow charts. This is done on a job by job basis.

- Integration

  The job stream flow charts are consolidated enterprise-wide to separate the application data flows (several independent data flows may use the same VSE label and even the same VSE disk space, but they become separate data sets in OS/390).

  In the Cortex tool a function called file integration automates the classification of files and jobs. It also prepares reports that help determine if a file requires manual intervention. The information needing to be understood for these files and jobs includes:

  – Where does the file or job reside?
  – Is it permanent or temporary?
  – Where does it come from?
  – Where does it go to?

- File classification

  Based on data flow patterns and other functional attributes (organization, record length, and so on) files are classified according to their life cycle:

work, cataloged temporary, handoff, backup, transmit, master sequential, master VSAM, and so on. File classification is a large JCL-related task done to define the life cycle of all of the data sets. This task does not have a high degree of difficulty but typically involves thousands of files. Each file must be classified and this can be very time consuming.

- OS/390 JCL Generation

  JCL includes, JCL procedures and inline JCL streams are generated according to standards defined for the new OS/390 environment:

- File Migration

  VSE data file attributes (record length, label, and so on) collected during VSE JCL conversion are combined with OS/390 data file attributes determined during OS/390 JCL generation, to generate VSE and OS/390 JCL streams used to migrate VSE files to OS/390.

Each step produces a number of error and warning messages which are systematically reviewed using specific CORTEX MS statistical analysis tools.

32.5.3 Phase 4: Initial Trial Conversion

The first mass conversion is the ***Initial Trial Conversion*** and occurs before custom modification of the mass conversion tools. It is used for analysis, rather than for generating OS/390 application material. The following mass conversions are trial mass conversions, which deliver OS/390 test application material with an increasing quality, as project and conversion tool customization progress. The first trial conversion, and all following trial conversions, simulate the actual, that is, the final, mass conversion.

The initial conversion is a conversion of a small, but representative, subset of your VSE applications, usually involving at least part of your most important work. This trial conversion lets the conversion team practice a conversion and verify their understanding of the conversion tool. The first test is different and takes longer than all the others. It may take two weeks. The last conversion may take a day. The first is the longest because more problems are discovered. Problems encountered during the first trial conversion are used to identify and document additional conversion tool′s modification requirements. The conversion tool′s custom modification is refined accordingly.

The subsequent conversions, called ***trial conversions*** go on for approximately six months with a conversion being performed every three weeks. It is an iterative process throughout the six months. Trial conversions deliver OS/390 programs, load modules, JCL streams, and files for regression test in the OS/390 environment.

In trial conversions the conversion tool has been customized, the first supply is taken and the JCL and the programs are converted. The next step is to look at it and see if it worked. Is the outcome what was expected? This is initial testing. This is the testing that occurs before the teams are brought in. Many problems can be expected at this stage.

Mass conversions require several JES2 initiators and are CPU intensive. They are submitted, as much as possible, at night or during weekends in order to avoid conflicting with daily operations if any other operations are sharing the same OS/390 system.

The first trial conversion starts with a complete fresh supply of the VSE conversion inventory. Every three to four weeks, the mass conversion starts from a fresh copy of the entire conversion inventory, in order to take into account the last VSE maintenance modifications. Between two supplies, additional mass conversions may be executed from the same supply, in whole or in part, in order to take advantage of the latest custom modification improvements.

The first trial conversion is complete when all OS/390 programs, load modules, and JCL are available for OS/390 tests.

32.5.4 Phase 5: OS/390 Regression Tests and Repeated Trial Conversions

The goal of these tasks is executing the converted jobs in MVS to verify that they function as they did in VSE, and to correct software conversion errors or file migration errors that generate test exceptions. Also during this phase full size copies of the VSE production will be migrated to MVS, and in the process will test the file migration procedure that will be used later on for switchover rehearsal and actual switchover.

Objectives of testing

1. To develop a broad base of knowledge as soon as possible.

   A major resulting benefit of the testing phase is the experience and training that is gained in the OS/390 environment. This is an excellent source of hands on training in a soon to be real life environment. The testing environment needs to include the use of the job scheduler, SMS, tape manager and Report Manager.

   Problems need to be identified as early as possible in order to provide the proper lead time for their resolution. The by-products of thorough testing are education, training, confidence, and success.

2. To make the applications fail.

   Testing is the process of working through execution time errors using the applications, jobs and data you have converted and moved into the new environment. Thorough testing and the elimination of problems are crucial to a successful conversion.

   Failure is good in the test environment; failure is not good in the production environment. Therefore, the better the design and execution of the test cases, the less chance for production failures.

Phases of testing

There are typically three phases of testing associated with the test cycle for applications. They are:

- Unit Testing
- System Testing
- Parallel/Simulated Production Testing

OS/390 online application tests include:

- Initialization tests which start each transaction to verify that the initial screen comes up
- Unit (or technical) tests are tests on a representative sample of transactions
- System (or Functional) tests are scenarios of chained transactions corresponding to application flows

- Simulated production (or acceptance) tests: in conjunction with batch production tests
- Network and performance tests with actual connection to future end users

OS/390 batch application tests include:

- Unit (or technical) tests: on a representative sample of jobs
- System (or Functional) tests are scenarios of chained job executions corresponding to application flows
- Simulated Production/Parallel (or acceptance) tests: replicate one day of VSE operations in MVS

Batch and online tests are coordinated to test the integration of batch and online applications.

The three phases of testing are building blocks. Each uses the output and successes of its predecessor to build on. Although they build on each other in a sequential manner, the boundaries between each in practice are blurred by overlapping test start dates and durations. Testing activity increases through this incremental building process from unit to system to parallel testing. The testing process is completed through a series of test switchovers which end with the final switchover/cutover. This last switchover converts your entire production system to OS/390.

32.5.4.1 Responsibilities

In a migration project it is the responsibility of the customer to do and resolve the testing of their converted applications. It is not realistic to expect that a contractor or service provider can come into the customer environment and understand the applications and application flow. It is the customer who understands how all these elements work individually and together.

32.5.4.2 Recommendations

Testing Priorities

All ″critical″ applications should be targeted for testing with daily applications tested first, followed by weekly, then monthly. Testing of quarterly and other periodically scheduled programs and applications can be scheduled after the production cutover, if necessary.

Test the most critical, or hardest to fix, applications first (that is, those programs, applications most subject to failure or degraded performance) thereby allowing more time to modify or tune these applications, if necessary.

Personnel Involvement in Testing

For each test implementation task (scheduling, setup, submission, execution control, result review and validation, debug) it is critical to use the exact same personnel who will be responsible for that task under OS/390 after switchover. Testing is not only to identify and correct conversion created regressions. It is the primary method to train and prepare the staff for OS/390 operations.

- **Application personnel** should be made responsible for providing test data, and for evaluating, and approving application test results. (Obtaining test data can be a problem - early plans should be made to define and obtain test data.).

  Involvement of applications to some degree in the test process can prove beneficial at OS/390 switchover and initial OS/390 production.

- **Operations personnel should do the operating during testing**. This is a vital part of their training. Operations personnel should be pointing out ease-of-use changes that can be made thereby improving OS/390 production jobstreams; that is, their feedback is important. They should also be preparing application ″Run Books″ during this time.

- **Systems programmer personnel** should only monitor system activity (for example, performance) and assist in problem resolution during tests; that is, they should not be running the tests.

- Online applications should be stress and performance tested prior to production.

  This type of testing involves bringing in (if on a weekend) as many users as possible to ″bang away″ at the terminals with as many different transaction types as possible. CPU-intensive batch, dump/restores, and other jobs that heavily load the system should be running during these tests, thereby better insuring good performance of online systems during peak system usage.

MVS Tools Testing

The tools selected for MVS operations (for example, tape manager, job scheduler, job preparation and report manager) should be used for the batch functional and parallel production tests in order to validate their installation and setup. If a job scheduler will be used in OS/390 production then use the same job scheduler during parallel testing.

DASD Requirements

Data migration during the testing phase creates a need for extra DASD. It is a far better scenario to overestimate your needs than struggle with insufficient DASD. Securing ample DASD can also be very helpful in reducing the time required for migration of databases. If the installation is growing in size it will generally not take long to make use of this capacity anyway.

Subsystem Storage Protect

It is recommended the implementation of SSP be delayed until after testing. The problem that can occur is that once implemented it can be difficult to identify the source of some problems and attribute them to testing old applications, to the SSP install or to a conversion issue.

32.5.4.3 Test Plan

Before any testing begins for any phase of testing the test group/team need to assemble and a test plan developed. Determinations need to be made about what actions will take place in each of the phases. Typical test plans turn out to be checklists that will be worked against. One team member will have the responsibility to sign off on each checklist item and signify whether the task was successfully completed or not. For those tasks that were not successfully completed this person would have the responsibility for the remedial plan. Each

test phase will have its own test plan. The key application development people at the installation must be involved when the test team is assembled.

Testing should not be something that just happens. Testing activities are an integral part of any migration plan and must be designed and controlled. When some plans are drafted, all too often, the word ″testing″ is all that appears on the task list or PERT chart. More thought and design should be devoted toward testing activities, so that proper resources and schedules can be allocated at the outset. Relative to testing, DP management has the responsibility to determine:

- What applications are to be tested.
- What constitutes a test (for example, parallels, data to be used, and so on).
- What audit information is necessary (SMF accounting information, operator logs, and so on).
- What acceptance criteria is required to show proof of success (timings, output compared magnetically, and so on).
- Who has the completion sign-off responsibility.

Jobs selected during the specifications phase, typically those scheduled to run within four weeks after the actual conversion and switchover, are regression tested in the OS/390 environment. OS/390 regression tests require careful planning and organization, OS/390 machine time and disk space, full access to VSE production procedures and documentation, availability of data and criteria to validate test results and direct participation of the customer. Problems discovered during the OS/390 regression tests are analyzed both in nature and in frequency. Typical solutions for problems with multiple occurrences involve improvements to the automated mass conversion process (conversion tool′s custom modification) followed by new trial mass conversions.

The OS/390 regression tests and repetitive trial conversions are organized into an interactive loop of tasks including:

- Orientation to the regression test phase
- Defining a regression test plan with scenarios for batch job scheduling and for execution of online transactions
- Defining a procedure to verify batch and online test results
- Executing online transactions
- Preparing and executing batch jobs
- Verifying and validating test results
- Identifying and analyzing test exceptions
- Applying short-term solutions to promptly resume the tests in progress
- Reviewing the nature and frequency of test exceptions
- Designing and developing permanent solutions based on the improvement of the automated mass conversion process
- Refining the conversion tools custom modification
- Supplying fresh copies of the conversion inventory for each new trial conversion
- Performing trial mass conversions
- Supplying new OS/390 JCL and programs for test in OS/390

- Migrating copies of VSE production files, needed for regression tests, to OS/390
- Migrating copies of VSE production databases, needed for regression tests, to OS/390

The conversion project team meets regularly to review the progress and status of the OS/390 tests. OS/390 tests typically take three to five months. The OS/390 test phase is complete when scheduled tests have been successfully performed.

32.5.4.4 OS/390 Automated Operations Tools

The objective of this task is to populate the OS/390 job scheduler, report/output manager and tape manager with job scheduling instructions, report/output management instructions and cataloged data coming from VSE.

Batch and online tests are coordinated to verify the integration of batch and online applications. The tools selected for OS/390 operations (for example, tape manager, job scheduler, job preparation and restart systems, EDI, and report manager) are used for the batch functional and production tests in order to validate their installation and setup.

If a report manager was used under VSE, the VSE report manager rules are migrated to the OS/390 report manager either from the VSE job scheduler (if applicable) or from operations manuals. During JCL conversion, the JCL-managed reports are retrieved and identified with unique report-ids. Their attributes (form number, number of copies, FCB, remote destination, output class, and so on) are collected into a file used to load the OS/390 report manager. The OS/390 JCL is generated free of report management attributes but containing the report-ids instead.

Active VSE tape files are the files that will be read after switchover by the generated OS/390 JCL. Inactive VSE tape files are all the other files cataloged to the VSE tape manager.

Once identified through JCL conversion, active tape files are either:

- Copied with reformatting (new OS/390 data set name and attributes) during switchover, if time allows, to an OS/390 tape volume, and cataloged to the OS/390 catalog and OS/390 tape manager, or
- Cataloged with a ″marking″ (for example by using a dedicated generic UNIT name like ″VSETAPE″) to the OS/390 catalog and OS/390 tape manager. A job service step is inserted at the start of each job to identify marked input tape file, if any, to modify the OS/390 JCL so that it can read the VSE-created file with its old VSE attributes and name. It takes only a few OS/390 production cycles for all VSE-created tape files to become obsolete and be replaced by a new OS/390-created version. Most VSE-created tape files have disappeared within a month after switchover, after daily, weekly, biweekly and monthly jobs have read the last VSE-created version of the file.

The list of inactive tape files (files not connected to the JCL converted to OS/390) is manually reviewed to identify, long before switchover, the legal archive and other files that do need to be migrated to OS/390. Those files can be copied to OS/390 tapes during the weeks and days before switchover, or left on VSE tapes and cataloged as is to the OS/390 catalog.

The large number of inactive and non-critical or obsolete tape files is not migrated. They are either eliminated, or the VSE volumes are set aside together with a last copy of the VSE tape file catalog at switchover time.

Any automated operations product, which will be used for actual production after switchover, must be installed, loaded with data used in production mode during regression testing, not for sample unit testing, but starting with full-size application functional testing and definitely throughout simulated production testing (see regression testing below).

Designing implementation standards for an OS/390 job scheduler or report manager, and loading them with job scheduling or report management instructions, is best performed with the on-site assistance of a specialist acting as a mentor and a tutor. Such a specialist may either be provided by the vendor (part of the product licensing agreement) or hired outside for a couple of months. The basic education included in the license agreement brings some understanding of the product features and limited hands-on practice, but it doesn′t replace the experience of a true specialist when it comes to full size implementation as experienced in a mass conversion from VSE to OS/390.

32.5.5 Initialization Testing

Initialization tests are performed to verify that online applications (tables, transaction programs, screens, and files) have been properly defined to CICS. Transactions are started to verify that the screens come up as expected. Initialization test scripts identify the minimum input required to get from a screen to the next one. At this level of testing, results and file updates are not verified.

This is a time to ensure the basics work. Many problems can surface here including mistakes in setting up the CICS environment. Frequently terminal definitions or file definitions need to be changed. During this process you are testing both your applications and CICS itself.

32.5.6 Unit Testing

Unit testing is the first phase of testing. Early unit testing is done using a sampling of jobs and applications. These should be a representative sample, a good mix of your jobs and applications used to expose problems. The idea behind the approach is that you will discover some problems. These early problems will be global problems that affect all your jobs and applications. You will go in and fix these problems and not rediscover these as you test other areas. If you tested everything at once, you would discover and rediscover the same errors over and over again.

An approach for testing that works in most migrations is to separate the test environments into two distinct areas. These two areas are online CICS and batch applications. Typically the online portion is tested first. The online testing is easiest because there is limited JCL conversion involved. Online testing is also unique because it is converted once and you are through with it.

32.5.6.1 Online Unit Testing

Prior to this task, an online test plan, and possibly detailed test scripts, have been developed with and presented to the test team, together with organizational instructions, during a pre-test orientation session.

CICS testing is an ideal place to begin the testing process. At least in theory, CICS transactions are fairly transparent between OS/390 and VSE/ESA. Assuming that the migration environment has been set up per the instructions in Chapter 25, “Prepare the Migration Environment” on page 401 in this book, CICS is the first area to be scheduled for testing and can occur very early in the project.

The online unit testing results that are sought during this early testing are:

- Get a screen to come up
- Travel within the screen
- Query a customer record

The testing continues with thorough sample unit testing of 30 to 50 representative transactions. The idea is to use limited resources (small test team and DASD) to verify that the conversion specifications and the automated conversion process generate sound OS/390 application material (programs, screens and files), which execute without ending abnormally. Results and file updates are verified. The conversion specifications and automated conversion process usually need some adjustment before full-size resource-consuming functional testing can start.

32.5.6.2 Batch Unit Testing

Prior to this task, a batch test plan has been developed with and presented to the test team, together with organizational instructions, during a pre-test orientation session.

Batch application tests start with sample unit testing of 30 to 50 representative daily jobs. The idea is to use limited resources (small test team and DASD) to verify that the conversion specifications and the automated conversion process generate sound OS/390 application material (JCL, code and files), which execute without JCL error, abends or abnormal return codes. Report contents and file updates are not verified at this stage. The conversion specifications and automated conversion process usually need some adjustment before full-size resource-consuming functional testing can start.

32.5.6.3 Data Migration in Unit Testing

The topic of data migration is implied throughout all the phases of testing. In unit testing transferring a couple of tapes of CICS data sets to MVS is sufficient for testing. By the time you are performing test switchover dress rehearsals you are copying data so that it matches exactly on both systems. In this way, data migration and the testing phases are synchronized events. Discussions on data migration will be included in each of the testing sections.

32.5.6.4 Timing between Online and Batch Testing

Normally online unit testing will be completed early in the project. From a project perspective, the focus frequently remains on the online workload and the system testing phase of on-ine is started right after the unit testing ends. This is because it is frequently too early in the project to begin batch unit testing. The batch testing is more complicated than online due to the JCL conversion. This is

the beginning of staggered and overlapping testing. Online system testing may overlap batch unit testing.

32.5.7 System Testing

In system testing the goal is to go through the online applications and verify that the outputs from each operating system are similar. In the systems test phase each application is tested in more detail and each application's output is verified as being close to the production system.

Online

Online application tests continue with full-size application functional testing involving a joint effort of the conversion team, applications support staff and a few selected end users willing to cooperate. User exceptions are filtered and corrected by applications support. Test exceptions created by improper conversion are identified and passed over to the conversion team for resolution. Temporary manual by-pass corrections are applied to the OS/390 application, in order to quickly resume testing. But each conversion error is also fully analyzed (qualitative and quantitative analysis), and a permanent and global (all occurrences) solution is developed by refining the custom modification of the conversion tools, or by developing additional conversion tools. Additional mass or partial conversions generate new and correct OS/390 application material. Per design, the test of online transactions often requires the execution of batch jobs. Both batch and online tests are integrated at this point.

Batch

The objective of this task in batch is to execute the converted jobs in MVS to verify that they function as they did in VSE, and to correct software conversion errors or file migration errors that generate test exceptions.

Full size copies of the VSE production files are migrated to MVS, and in the process, the file migration procedure to be used later on at switchover, is tested.

In system testing you are testing the code more as an application. This is where JOB A feeds JOB B which feeds JOB C and so on. It may be general ledger or accounts payable. At this time a key task is ensuring that all the input and output are correct.

Batch application tests continue with full-size application functional testing involving a joint effort of the conversion team, production control and applications support staff. User exceptions are filtered and corrected by the production control or applications support. Test exceptions created by improper conversion are identified and passed over to the conversion team for resolution. Temporary manual by-pass corrections are applied to the OS/390 application, in order to quickly resume testing. But each conversion error is also fully analyzed (qualitative and quantitative analysis), and a permanent and global (all occurrences) solution is developed by refining the custom modification of the conversion tools, or by developing additional conversion tools. Additional mass or partial conversions generate new and correct OS/390 application material. Per design, the test of batch applications often requires the execution of online transactions. Both batch and online tests are integrated at this point.

One problem that can surface in this phase of testing is with data set definitions and the use of temporary data sets. A problem can surface when the data from

the third job in the sequence is missing due to being treated as a temporary data set.

The management of transition files is very different between VSE and OS/390. In OS/390 there are only a couple of possibilities. In VSE it depends on the type of organization, the products used and the traditions followed.

These problems don't show up until the applications are tested together. This is the right time for these to surface. If these all occurred during parallel testing it would take a very long time to complete.

32.5.7.1 Data Migration in System Testing

The system test phase may or may not require that more data is available but frequently requires that more accurate data is needed. An example is working with databases that are two or three months old. In some cases these applications will not run because they have been written to work only on recent data. Recent data may be required. Another hurdle can be that these applications have dependencies on batch data feeds to them.

32.5.8 Parallel/Production Simulation Testing

Parallel testing is more representative of what the system will be like in OS/390. In parallel testing you start with a day, typically a Sunday, and then do one day at a time. The data from each day is scrutinized. Here the end user community will get involved in testing. Output reports are reviewed and verified that they are what they should be. This depth of review and comparison also requires that in parallel testing you start to synchronize the data between systems.

At this point your network should provide access to this environment from any terminal. This is an important consideration in allowing quantities of end users on the system who should not be restricted to a small number of terminals with access to MVS.

Parallel (batch) production tests will include all jobs executed for a period-end day. Duplicating VSE execution will not always be possible, due to the integration between online and batch applications and some side effects of job execution date.

The final phase of parallel testing is ensuring the output is the same from both systems, where actual production is compared to the test output. In this phase there are two systems running in parallel. Monday is run on the OS/390 system and compared to Monday's output and reports from the VSE production system. For week ending, month ending and year ending scenarios that actual date change becomes the test.

32.5.8.1 Data Migration in Parallel Testing

In parallel testing the data in the systems needs to be synchronized as you will be comparing production runs under VSE with test runs under OS/390. Typically by this time you have done switchover rehearsals a couple times. Now just before parallel testing begins you will synchronize the data for a particular day, for example, Monday or Tuesday.

Job Simulation

The goal is to get through as many days as possible. It is a comfort to know at switchover, that a week′s jobs can be processed. There are significantly fewer monthly jobs than weekly jobs. There are significantly fewer yearly jobs than monthly jobs. Plan the cutover to not occur at month end.

The final phase of parallel testing is ensuring the output is the same from both systems, where actual production is compared to the test output.

Date Concerns during Parallel Testing

Date concerns at this stage should not be as large a concern because of synchronizing data for particular day. Be certain that if the output reports run clean it is not due to them being empty as a result of not finding any date specific data. Also results can be a bit unpredictable when jobs are run with dates that are unexpected. Be aware of the age of the dates. If necessary the dates can be set externally through the JCL.

32.6 Implementation Phases

The implementation phases are typically the two weeks that precede the actual switchover. Once testing indicates that the conversion process is working, your installation is ready for switchover. After the final mass conversion, switchover typically requires only a single weekend. The switchover process consists of backing up the current VSE environment, switching files to OS/390, priming the MVS catalogs, and initial MVS production operations. This is the period when the VSE system is frozen. You do not want to be making changes to the VSE system during these two weeks.

The project phases during the Implementation Phases are:

- Actual Conversion and Switchover
- Initial OS/390 Operations

The final and actual mass conversion will be started after MVS tests have been successfully completed. It will deliver the actual MVS production material. The actual JCL conversion may be scheduled one or two weeks before the actual program conversion, in order to apply final manual JCL modifications. There will also be a special one-time translation of all applications′ development source code, but without any compilation or JCL generation.

These phases include the following key tasks:

- Mass Conversion of Development Source Code
- Final JCL Mass Conversion
- Final Program Mass Conversion
- Initial OS/390 Operations

References for additional conversion information can be found in:

- Appendix B of the *MVS-MS Planning Guide* named ″Sample Switchover Plan″.

32.6.1.1 Converting the Development Material

This is the code that the systems programmers are working on. It is recommended that the conversion of these materials take place as early as possible. This conversion is not normally done at switchover time. It is not production material.

This task generally coincides with the transfer of a significant number of application developers to the OS/390 platform. Development under CMS or VSE can continue to a point. Convert the work in progress and then move the people to OS/390. This can occur three months before switchover all at once or be moved in a staged approach each week. How these are moved depends on the amount of development activity in progress, what stage of development it is in, how much growth is happening and what machine resources are available. Converting these materials and personnel to OS/390 before switchover also provides extra experience in the environment.

32.6.2 Phase 6: Actual Conversion and Switchover

The actual conversion is the final mass conversion, starting with a final fresh supply of the entire conversion inventory. Both actual conversion and switchover are performed within two to four weeks, with the actual conversion starting on a Monday, and the switchover being completed on a Sunday. Production testing is used to validate the actual conversion before switchover. A final supply of conversion inventory, a few days before switchover, is used to identify any late VSE change control and to carry it over to the converted OS/390 application.

As opposed to trial conversions, the actual conversion is followed by a mass migration of all permanent VSE production data files and databases to OS/390, which requires a short production outage during the weekend (typically on early Saturday morning). After the file migration, some (mostly weekend) jobs are executed and some online applications are started and verified in actual production mode.

During the month preceding the switchover, all parties participate in planning and preparation activities. These activities produce the final file and database migration JCL streams and switchover task lists.

The actual conversion and switchover are complete when the scheduled production jobs and online applications have run successfully in the OS/390 environment.

The key elements of preparing for the actual conversion are:

- Final JCL Conversion
- Final Program Conversion

32.6.2.1 Final JCL Conversion

A key task associated with the final JCL conversion is freezing the production database. The actual JCL conversion should be done as late as possible before the switchover, but before the final program conversion.

Another key task is performing the known manual changes to the production database. At this point, the production database is established in a final form, so further modifications made to jobsets during VSE production must also be made to the conversion tool production database. Implement a change control

procedure in order to apply jobset maintenance concurrently with maintenance to VSE production.

32.6.2.2 Final Program Conversion

There are two key tasks associated with the translation and compilation of all the VSE source materials for the final program conversion. First is the final mass translation with the objective to translate all of the source modules in their current state, overnight, automatically, and without errors. Second is the mass compilation for the final conversion. The objective of this task is to compile and link-edit all of the translated source modules overnight, automatically, and without errors.

32.6.3 Switchover

The switchover weekend may not be a typical weekend, because it is a period when operations are interrupted for the switchover. This interruption may be for as little as a few hours or for as much as one or two days. Before switchover, the production workload runs under VSE. After switchover, the production workload runs under MVS.

The objective here is to switch VSE production over to MVS and continue operations under MVS.

This transfer must be:

- sufficiently fast to allow mass switchover to take place within the allotted time
- in conformity with the generated MVS JCL to avoid modification during the initial runs
- complete, because you should avoid coexistence of VSE and MVS systems.

In addition, backup operations on the VSE side are necessary before file transfer to provide for a possible return to VSE in an emergency or for a recovery of overlooked files. Backup operations on the MVS side are necessary to permit recovery, if required, during initial MVS operations.

Switchover weekend activities include terminating VSE operations, backing up the VSE and OS/390 environments, switching the network, executing file migration and cataloging procedures, starting OS/390 operations with CICS transactions and jobs according to normal weekend schedule, and supporting OS/390 operations through review and resolution of exceptions. On Sunday, the switchover is validated (stay under OS/390) or VSE fall back procedures are implemented (return to VSE in case of unexpected difficulties).

32.6.3.1 Data/File Migration

On switchover day you have to have all of your data available to your new system. This process of multiple dress rehearsals of switchover gives the whole team confidence about the final switchover. At that time the team has been through it many times.

32.6.3.2 Additional Switchover Tasks

These tasks may also need to be addressed during switchover:

- RJE workstation configurations
- NJE end users must change their JCL for job submission
- Configuration of PC workstations
- Migrate the tape manager to MVS
- Migrate the database manager to MVS
- Migrate the Report Manager database to MVS
- Secure on-site assistance from major vendors

Preparing for switchover takes about a month. A detailed and timed switchover plan is developed. Final switchover file migration and backup procedures (VSE and OS/390 JCL streams) are developed, starting from similar procedures used during testing. The OS/390 environment (catalogs, tape manager, disk space) is cleaned up from any trace of testing. A final supply of programs and JCL is compared to the supply used for the final mass conversion. Modified VSE elements are identified and converted to OS/390 (or the VSE changes are manually applied to the OS/390 version) to bring the OS/390 applications to current VSE production level.

32.6.4 Phase 7: Initial OS/390 Operations

The objective of this task is to support initial OS/390 operations with conversion-related issues. The conversion team works with the operations team to analyze and resolve any operations exception created by the conversion of programs and JCL, or by the file migration. In particular, the conversion team systematically verifies if the exception is isolated, or if similar cases can be identified and corrected before they create new operations exceptions. Initial OS/390 operations typically require 24-hour on-site assistance during the first week. It tapers down to 12-hour on-site plus 12-hour on call during the second week, and 8-hour on-site plus 16-hour on call during the third week. The need for operations assistance by the conversion team ends three to five weeks after switchover.

After conversion, start your OS/390 system slowly. Ease into it and segment it where possible. Don’t let the job scheduler run free. Bring up one CICS region at a time. Run the first batch job and check the results before proceeding. If it is bad you can still back out. Always allow for a way to return to VSE if you have to.

The assisted MVS operations phase typically lasts three to five weeks. This phase is complete when all daily, weekly and monthly production jobs have executed successfully within the MVS environment. This also marks the end of the VSE to OS/390 migration project.

Chapter 33. Conversion Services and Tools

The actual process of converting JCL and programs from VSE to OS/390 can be a very tedious, time-consuming and labor intensive set of tasks. Fortunately, there are tools available from both IBM and non-IBM sources to help automate most, if not all, of the conversion process. The purpose of this chapter is to discuss some of the more popular conversion tools. It should be noted that the tools discussed in this chapter only represent a portion of those currently available and in no way constitute a blanket endorsement simply by their mere inclusion in the chapter. Users are strongly encouraged to investigate ***all*** available tools and determine that particular tools' applicability based upon specific user requirements.

Users are also encouraged to contact as many of these types of firms as is possible and practical. At the time of publication it is our desire to publish a listing of service providers on the VSE/ESA home page. Therefore, the VSE/ESA home page should be consulted for the most current list of IBM business partners providing VSE to OS/390 migration

Also, visit the VSE/ESA Home Page for VSE to OS/390 migration information at:

`http://www.s390.ibm.com/vseservices`

for contact information.

This chapter discusses the following topics:

33.1 Conversion Services

33.1.1 IBM Global Services

IBM Global Services provides project management and migration services for migration projects.

- Call 1-800-IBM-4YOU
- Contact your IBM Marketing Representative or IBM Business Partner
- Visit the IBM Global Services Home Page at
  http://www.ibm.com/services/
  http://www.s390.ibm.com/vse/

33.1.2 Automated Migration Services (AMS)

AMS provides complete migration services using mass migration methodologies based on the Cortex MS tool. AMS provides on-site migration and conversion services primarily in the United States. AMS personnel were involved in the original development of the Cortex tool and have continued to develop its capabilities. AMS will also perform the phase one project as a standalone service.

AMS is an IBM Business Partner

- Call 1-800-482-6267 or,
- Contact AMS through IBM

33.2 Conversion Tools

33.2.1 VSE/ESA Facilities

VSE/ESA 2.3 added a new function useful for developing an inventory of a VSE/ESA system's jobs, including the so-called hidden JCL found in standard label areas and in standard assignments. The VSE/ESA JCL Analyzer is included in ICCF library 59 as a sample source program that can be tailored to extend its functions.

Output from the JCL Analyzer is a Comma Delimited Text file, in a format suitable for processing by programs on PCs, such as Lotus' 1-2-3 spreadsheet or the VisualAge Application Understanding tools running on OS/2 or Windows NT. The Application Understanding tool can display a graphical analysis of the JCL job stream.

You can find further details at the following Web sites:

```
http://www.software.ibm.com/ad/va2000
http://www.software.ibm.com/ad/cobol
```

For an overview, see *VSE/ESA Enhancements Version 2 Release 3*, SC33-6629. Note that this facility is available via PTF for VSE/ESA 1.4, 2.1, and 2.2, as well as 2.3. Member ARDWREAD in ICCF library 59 is a detailed description of the JCL Analyzer. Sample jobstreams are provided along with the programs required.

33.2.2 IBM OPTI-AUDIT for VSE

While the IBM OPTI-AUDIT product was originally developed for performing an assessment of Year 2000 readiness, it is also an excellent tool to use when performing the similar set of activities associated with a VSE to OS/390 conversion; that is, taking an inventory and assessment of the programs, data sets and JCL that will be converted.

IBM OPTI-AUDIT for VSE Version 1.1.0 is a new program offering that enables you to perform a static analysis of VSE libraries and build an inventory of all programs contained on your VSE system. This is a vital first step in your year 2000-readiness planning. IBM OPTI-AUDIT is a powerful tool designed to assist you with your migration from an earlier VSE SP or ESA system to a current VSE/ESA Version 2 system. It can also be used when converting from VSE/SP or VSE/ESA system to OS/390.

IBM OPTI-AUDIT for VSE monitors the execution of batch jobs extracting job/program/file cross-referencing information. Use IBM OPTI-AUDIT as a source scanning facility on COBOL source code.

IBM OPTI-AUDIT for VSE produces a variety of reports to manage the year-2000 conversion process, including:

- Program status
- Program usage
- File cross-referencing
- Job cross-referencing

33.2.2.1 Product Highlights

IBM OPTI-AUDIT for VSE Version 1.1.0:

- Captures and builds an inventory of all programs running on your VSE system
- Monitors the execution of batch jobs extracting job/program/file cross-referencing information
- Provides a source scanning facility for COBOL
- Produces a variety of reports for the year-2000 conversion

33.2.2.2 Product Details

- Performs a static analysis of VSE libraries.
  - Libraries not required can be excluded
  - Creates a central repository (database) of PHASE related information, such as, phase name and usage tally.
- Monitors the execution of batch jobs marking the phases as ′active′ and logging phase, job and file (for example, data set) cross-referencing information.
- Functions available through batch job submission:
  - Flag phases as ′Y2K READY′, including the facility to remove this setting.
  - Scanning of COBOL source code for date related information
    - Output is divided into four reports:
      - REPORT 1 - Suspect Verb analysis (looks for date suspect COBOL verbs, for example ceedate..)
      - REPORT 2 - Suspect Variable Analysis (looks for common variables used as date fields, for example, 9(6), and X(6))
      - REPORT 3 - Suspect Variable Scan (scan for USAGE of variables identified in reports 1 and 2)
      - REPORT 4 - Generic Search Results (uses a software supplied table of character strings and returns matching lines of code) User-supplied variable names can be included or excluded from the reports as required.
    - Opti-analyzer -- generates an analysis of a batch phase. It identifies each module called, providing a statistical break-down of all supervisor calls. Specific date-related calls such as GETIME (SVC 34) can be identified.
    - Reports

      Database (phase) - four reports are available:

- REPORT 1 - ACTIVE phases (lists all ′executed′ phases by library).
- REPORT 2 - INACTIVE phases (lists all ′dormant or yet to be activated′ phases by library).
- REPORT 3 - Y2KREADY phases (lists all phases flagged as Y2K ready by library).
- REPORT 4 - DUPLICATE phases (lists all ′duplicate′ phases by library).

Log File - four reports are provided which detail PHASE usage:

- REPORT 1 - File Report by Phase (lists all ′active′ phases sequenced from most active (using a tally of the number of times the phase was invoked) to least active).
- REPORT 2 - File Report by Library (phase/library cross-reference by library).
- REPORT 3 - File Report by Jobname (phase/job cross-reference by job name).
- REPORT 4 - File Report by Program (phase/program cross-reference by program).

Log File - three reports are available which provide details on FILES (for example, data sets) used:

- REPORT 1 - File Report by Program (program/data set cross-reference by program).
- REPORT 2 - File Report by Data Set (data set/phase cross-reference by data set).
- REPORT 3 - File Report by Jobname (job/data set cross-reference by job name).

33.2.3 IBM COBOL and CICS Command Level Conversion Aid (CCCA)

The tedious process of COBOL migration can be easier than you think with the IBM COBOL and CICS Command Level Conversion Aid (CCCA) for VSE Release 1. CCCA for VSE, a Program Offering, helps you easily convert old COBOL source code and copy books to the new COBOL standard.

CCCA for VSE converts OS/VS COBOL, DOS/VS COBOL, and COBOL 74 Standard VS COBOL II (either VS COBOL II Release 3, or VS COBOL II Release 4 (CMPR2)) source code to COBOL 85 Standard VS COBOL II Release 3 or 4 (NOCMPR2), or to IBM COBOL for VSE. You can customize this conversion process to meet your unique requirements.

CCCA simplifies the migration process by:

- Identifying and converting source code
- Reducing the effort to convert programs
- Minimizing conversion errors

33.2.3.1 Product Positioning

COBOL and CICS Command Level Conversion Aid for VSE Release 1 is positioned as a COBOL migration aid designed to provide:

- Automated conversion of most COBOL syntax differences.
- Programmer productivity for the migration process.
- Reduction of manual conversion errors.
- Flexibility through an open converter design.
- Generation of conversion management reports.

CCCA eases the migration process, allowing customers to upgrade their old COBOL technology, to the new COBOL technology (as in VS COBOL II) quickly. Once on the new VS COBOL II product, customers are positioned for upcoming technological advancements, such as object-oriented technology and client server.

33.2.3.2 Technical Description

As in CCCA Release 2 for MVS and VM, COBOL and CICS Command Level Conversion Aid for VSE Release 1 contains the following components:

- Language Conversion Programs (LCPs)
- LCP Compiler
- Driver
- Report Programs
- Front-end Panels
- Tables

Language Conversion Programs (LCPs): The LCPs are written in a COBOL-like language used to describe the process of converting the differences between the old COBOL language (that is, OS/VS COBOL, DOS/VS COBOL or ANSI 74 VS COBOL II), and the new COBOL language (that is, VS COBOL II ANSI 85 Standard or IBM COBOL for VSE).

A set of LCP modules describing how each old COBOL language element is to be converted into the new COBOL language is provided with this tool. The set also provides CICS command level-related statements conversion. The basic set enables users to convert most differences, and can be very easily customized for specific conversion requirements.

By adding new LCPs, the user can:

- Change COBOL source syntax.
- Add, replace or remove a word, clause or statement.
- Indicate where the newly generated COBOL source needs to be reviewed for possible manual action.

A set of panels is provided to help the user with LCP development within CICS.

LCP Compiler: Each LCP module is processed once by the LCP compiler component and is then used by the driver component to convert each statement requiring conversion. The basic set of LCP modules included in the product is already processed by the LCP compiler.

Driver: This component reads the COBOL source program, extracts copy members from the input source file, and executes the conversion process according to the corresponding compiled LCPs.

The driver produces four types of output:

- New COBOL source code in new source library (optional).
- New COBOL copy module in new copy library (optional).
- COBOL source statement diagnostic listing.
- Conversion management report data.

The diagnostic listing is a statement-by-statement log showing the result of the conversion process.

An analysis is made to ensure that user-defined names do not conflict with words newly reserved for the new COBOL language. Where conflicts appear, a two-character suffix is appended to the user name. This suffix can be modified by the user.

Conversion management report programs can be executed on request to provide the status of each converted program, and tell when it was converted, and whether user involvement is required. A ″where-used″ list of copy modules, files, and CALL statements can also provided. These reports provide an excellent cross reference.

33.2.4 SISRO - CORTEX-Migration System (CORTEX-MS)

SISRO designs, develops and commercializes computer automation software products which make it possible to drive and monitor computer systems, while planning processing and managing data.

The CORTEX and JobServe products (for mainframes and distributed environments, respectively) provide solutions that are modular, user-friendly and based on innovative technology, which is a guarantee of openness.

SISRO has nurtured close partnerships with many leading actors in the computer market, notably with Microsoft (Solution Provider since 1992), IBM (S390/PID and SDP Associate), DEC (Member of ASAP program), and Oracle (Member of Oracle Value Service Program).

SISRO′s software packages are marketed worldwide via a network of distributors and integrators, and through subsidiaries in Europe and the United States.

The CORTEX-MS product uses the mass migration method. Mass conversion automation is the result of six software components designed for conversion of VSE production applications to OS/390. All the components are menu driven through TSO/ISPF panels.

For more information contact SISRO at

1. website www.sisro.com, or
2. by phone in the US at 919-460-9870.

33.2.5 Computer Associates

33.2.5.1 CA-Convertor

CA-Convertor is a comprehensive system to manage and implement the conversion from VSE to MVS operating systems. CA-Convertor converts VSE COBOL and Assembler language programs and VSE JCL to true MVS programs and JCL, while automatically creating applicable documentation. It performs the burdensome conversion tasks that, under manual control, would be error-prone, tedious and repetitious. With CA-Convertor, the entire process is automated, eliminating redundant data manipulation and unnecessary manual procedures. Programmer productivity levels are not affected by constantly monitoring the conversion and, as CA-Convertor can execute on both VSE and MVS, you can start the conversion without requiring a production MVS environment.

Available for: MVS, VSE

33.2.5.2 CA-DUO

CA-DUO is VSE under MVS transition system software designed to simplify the conversion from the IBM VSE environment to MVS. CA-DUO provides the interface that allows existing VSE application programs to run directly under MVS without source program alterations. This unique approach conserves considerable data center resources, and facilitates a true MVS production environment in the least possible time.

CA-DUO allows all MVS facilities and most VSE facilities to be utilized. It uses standard MVS JCL, which may be generated by CA-Convertor. CA-DUO supports the following programming languages, access methods and data management system: Assembler, COBOL, FORTRAN, PL/I, RPG, RPG-II, DAM, ISAM, VSAM, EXCP, QSAM, BSAM, BISAM, QISAM, DL/I, BOMP, DBOMP, TOTAL, ADABAS/DB, CA-IDMS/DB and CA-Datacom/DB.

Available for: MVS

33.2.6 The Source Recovery Company

The Source Recovery Company recovers missing COBOL or Assembler source code from MVS, VM or VSE executable modules.

The recovered source is guaranteed to produce an executable module that is 100% functionally equivalent to the original executable module that you send them.

The Source Recovery Company can help save a great deal of time, effort, cost and minimize your overall Year 2000 and conversion project risk. You can avoid an expensive rewrite of a program, or even more expensive replacement of an application.

Listed is a brief summary of the services provided by The Source Recovery Company:

33.2.6.1 Recovery/SRC

This is the basic service provided by SRC. The basic recovery utilizes a proprietary technology that generates the source code from the load module supplied by the client. Data names and labels within the programs recovered are generic.

33.2.6.2 Rename/SRC

This is a service that is sold in addition to the basic recovery service (Recovery/SRC). The renaming service matches client-provided copybooks to the data definitions generated by the recovery technology. The source returned to the client will contain the original data names as defined in the copybook where appropriate.

33.2.6.3 Reconcile/SRC

This service is sold in addition to the basic recovery service (Recovery/SRC). The reconciling service blends client provided versions of the source program with the program generated by the recovery technology. The source returned to the client will contain appropriately matched data names, labels and original comments based on the client provided version of the source program.

33.2.6.4 VersionMatch/SRC

This service is offered separately from the other recovery services. The purpose of VersionMatch/SRC is to match source code to load modules. SRC will respond with a written report that identifies which version of source matches the load module. If no match is found, the client is afforded the option of requesting SRC to provide additional recovery or reconciliation services as defined above.

The Source Recovery Company can recover code written in any version of IBM Assembler or COBOL. Part of the process allows for discovering in just which version of these languages the source code was originally written, and they will return it to you in the original ″flavor″ of that language. In fact they are always looking for older versions of compilers, and if you have one, they′d like to hear from you.

33.2.6.5 A COBOL Recovery Example

Since the compiler does not store any of the original data or paragraph names in the object module, Recovery/SRC must build generic names. The names generated are based upon where the item is located in the program (such as, ″WS″ for an item in Working Storage, followed by a hexadecimal number representing the item′s displacement from the beginning of the record). Rename/SRC and Reconcile/SRC are additional and optional services offered to enhance the actual datanames in the recovered source code.

33.2.6.6 Original Program Source Code Example

For information about source code recovery and an example of a program recovered by The Source Recovery Company in COBOL, see the Internet at:

```
www.source-recovery.com
```

Part 8. Migration Experience

Chapter 34. Customer Migration Example

This chapter describes an actual user experience with migration. Since every customer environment is unique, care should be taken when drawing comparisons, especially in areas of resource and capacity.

34.1 Background

Cust1 is a relatively fast growing company that evolved from a small 4341 to a 3-way 9672 in a little more than a decade. The I/S organization was constantly challenged to support new business with very little lead time. Their VSE operating system was always being pushed to its limits supporting the growing workload. The computer operations had grown to a near 21 shift operation (only a two to four hour window early Sunday morning for maintenance). For five or six years before the final decision to migrate to MVS, serious consideration was given to an MVS migration because of the constraints of the VSE architecture on their business, but new enhancements to VSE relieved the urgency. All of the measures to improve VSE performance were taken, for example, multiple VSE guests for parallelism, faster engines, faster DASD, VSE ESA features and so on. Each time a new piece of business was taken on, the I/S organization was concerned if they would be able to contain the workload, and usually suffered a lot of pain balancing resources. Finally it was decided that the move to MVS would be the only recourse to support the future business growth.

34.2 Environment

34.2.1.1 Hardware

- Bipolar air cooled 3-way processor
- The customer changed to a 9672 3-way in the middle of the project. This change was made primarily because of an IBM marketing package that actually provided more resource for a lower lease cost. This occurred at a time in the project when the extra capacity (mostly central storage) was very welcome (systems tests).
  - 256MB storage
  - 1GB on the 9672
- RAMAC DASD
  - Two racks of RAMAC I each 2/3 full (3380)
  - One rack of RAMAC II 1/3 full (3390)
  - All racks filled by switchover for testing bubble

34.2.1.2 Software

- VM/ESA
- VSE/ESA (three guests)
- The VSE level with turbo was installed before the project, but the turbo dispatcher was not enabled because of poor performance and availability issues.

34.3 Inventory

- 1500 COBOL programs - mix of DOS/VS COBOL and COBOL II
- 2600 RPG programs
- 80 Assembler programs
- 8000 JCL steps

34.4 Resources

In this project it was sometimes hard to get the various groups to focus on the migration when needed. This was because of other priorities with day to day problems and projects. This is not all that uncommon, especially in a growing organization. This however, needs to be monitored closely or delays in the project will result.

- Systems programmers
  - Three systems programmers on VSE. An experienced MVS systems programmer was hired at project start. One VSE/VM systems programmer worked almost exclusively on VM/VSE until near switchover (normal ongoing VM/VSE support). One other programmer split time between VSE and MVS. The new MVS systems programmer spent full time on MVS.
  - Two database administrators/programmers. Both split time between VSE and MVS.
- Application programmers
  - Two applications programmers were used with primary focus on the MVS project in the early stages of the project. These people would draw on other resources in their group as needed (for example, during testing). As the levels of testing progressed so did the number of application programmers required.
- Operations
  - Early in the project there was not much involvement from the operations group as the systems support group performed many of those functions. There was some involvement from some of the key operations people during planning phases. As the project progressed there was more involvement from operations.
- Management
  - A second line manager took the responsibilities of project manager. Although much of his time was spent on normal day to day responsibilities, the majority of his time was devoted to managing this project.
- Consultants
  - IBM Global Services teamed with Automated Migration Services for overall project management and to actually perform the migration. IBM also performed various systems programming tasks as required. In addition, ISV vendors were contracted during various phases of the project to perform customization/education of their products.

34.5 Duration

Due to the data sharing requirements, the availability requirements, and, in general, the dynamic environment of the business, it was decided the mass conversion method was the way to migrate to MVS. In the past years of rapid growth not much time was spent defining and enforcing systems management disciplines which resulted in uncertainty of what source code, phases, JCL and so on, were production and which were obsolete or test. Since these elements are the key indicators in determining the scope of a project like this it was decided to break the project into two phases. Phase one, the application inventory which through the use of software tools identified and helped segregate the elements that were used in production and the elements required for test. Phase two, which was basically the conversion of the programs and JCL, testing and switchover to MVS. As stated in earlier sections, the mass conversion method is the most common method to migrate to MVS. Also, the two phased approach is becoming more popular because of its many benefits, most importantly a much more accurate prediction of the actual cost of the project.

34.5.1.1 Phase One

About eight weeks was spent in this phase. First the MVS software required to run the tools was installed as a VM guest. This part can be eliminated by buying time on an existing MVS system to run the tools. Next, all of the VSE libraries required were imported to the MVS system and the tools run which resulted in a series of reports that showed exceptions (missing, unreferenced and so on). Representatives from systems support, operations and applications development departments reviewed the reports, made changes (move, delete, find elements) and resubmitted the new data to the MVS tools. This is an iterative process with the end result a fairly clean inventory.

34.5.1.2 Phase Two

This is comprised of multiple phases as described in earlier chapters. It took approximately thirteen months until switchover to MVS. This time was a couple of months longer than originally planned primarily for two reasons, inadequate testing in the final phases of testing and finding a weekend where the normal weekly four hour window could be increased to eight hours to accommodate the switchover to MVS.

34.6 Performance

As mentioned earlier VSE was run as multiple guests under VM/ESA. Once MVS was switched to production, it remained as a VM guest for a couple of months. After this MVS was run under an LPAR. VMPRF was installed on VM and was run daily to create summary history reports. The reports were not granular enough to break out CPU utilization by virtual machine, therefore, the numbers reflected the sum of the VSE and MVS guest while testing applications under MVS. There was a time when there was little or no activity on MVS while VSE production was running, mainly third shift. By comparing the third shift before switchover (VSE production) and third shift after switchover (MVS production) fairly comparable numbers could be obtained. One problem with this was that the time in the project when there was only VSE workload during third shift was when the 9121 was installed and after switchover the workload was on a 9672. The other variable was that there was no way to prove that there was an equal workload at the two times. However, using these rough comparisons and normalizing the CPU times using the LSPR ratios, the CPU utilization stayed

about the same after switchover, with MVS showing a couple of percentage points higher.

Since the workload at this installation is not the typical online on prime shift and batch on second and third, it was not as easy to make comparisons on a batch window length. Overall it appeared that throughput was much better. There were other performance benefits realized. A good example was a large database that affected VSE performance could only be reorganized (to improve performance) on holiday weekends because of the many hours it took to perform the reorganization. Under MVS this reorganization time was cut by two thirds allowing this function to be performed on a more regular basis. This reduction in time was attributed to the better throughput of MVS and in the internal design of the MVS version of the ISV database manager.

34.7 Benefits

Other than the performance benefits mentioned above there were many other benefits realized by going to MVS. As new business comes along MVS is able to absorb the workload without many of the problems previously encountered by the I/S staff when they had to try to ″fit″ this into the VSE environment. Examples of this are the ability to connect new NJE customers to one line using SNI as opposed to multidrop lines for 3270 access and a point to point bisync NJE line. When new business dictated more DASD, there was flexibility to go with the latest RVA DASD and not worry about operating system support issues. There is also the flexibility to grow the processor horizontally or vertically, which ever is more cost effective or quickest in order to take on new business. Even more important in this high availability environment where unscheduled VSE IPLs were frequent, MVS has run over a year without an unscheduled IPL.

Part 9. Appendixes

Appendix A. Education Information

The task of providing the right training, to the right people, at the right time, at the right location is a small project of its own. There are many variables to consider, including costs, not the smallest of which is simply determining if a particular course is available. A good training plan will be the right balance of these elements based on the needs of the installation.

The one thing that holds true for all migrating installations is that each installation has unique training needs. In addition to unique products and programs, the training and experience of operations, application development and systems programming personnel are different. The training plan will be different for a new hire than for a seasoned veteran or journeyman skill.

The major elements to be considered in planning for the education needed for your installation during a migration are:

- What training is needed and what courses are available
- When are courses scheduled and when are they needed during the migration
- Who will provide the training
- Where the training will take place

A.1 What Training is Needed and What Training Courses are Available

A.1.1 OS/390 Classes

Here are some of the key OS/390 classes from the IBM E&T Web page (http://www.training.ibm.com/ibmedu/roadmaps/mainframe/os390/):

- Introduction

  Introduction to OS/390
  OS/390 Facilities
  Fundamental System Skills in OS/390

- Operations

  System Operations for OS/390
  CMOS Complex Systems Availability and Recovery
  S/390 Hardware Management Console (HMC) Operations
  S/390 Multiprise 2000 External Support Element (SE) Operations

- Installation

  OS/390 Installation
  Using ServerPac to Install OS/390

- Other

  MVS Job Control Language and Utilities
  MVS VSAM and Access Method Services
  Transition from MVS/ESA to OS/390
  Automation Using System Automation for OS/390
  Measurement and Tuning for MVS/ESA Version 5 and OS/390

It is recommended that a class on TSO and ISPF, to help navigate through panels, be taken prior to the MVS Installation and SMPE class. The IBM SMPE class is a good and necessary prerequisite to the MVS install class. SMPE is similar to VSE MSHP. It provides key information on installing products and applying PTFs and is good for VSE systems programmers.

The focus of the MVS Installation class is for MVS customers installing a new version of MVS. There may be additional considerations for first time users. These needs may be addressed in customized training.

A.1.2 Custom Classes

IBM Education and Training (E&T) can help you with your training challenge by providing customized training. E&T specializes in presenting a common basic curriculum to a mixed audience of operators, applications programmers and system programmers (OS/390 basics, how to use ISPF, JCL basics) and then splitting each audience to learn those topics specific to their job performance.

The following training on MVS fundamentals is intended for individuals skilled and experienced with the VSE environment, but new to the MVS environment. After this series of courses, the MVS professional should understand the basics required for MVS.

A customized base VSE to OS/390 training plan would include the following:

- an overview of the components of OS/390
- a hands-on ISPF primer
- basic JCL and utilities preceded by a TSO primer
- OS/390 commands
- JES2 commands
- DIM (Data In Memory) techniques for applications programmers
- advanced OS/390 training for systems programmers

A.1.3 OEM Product Education

OEM product education must also be coordinated during the migration. Training on OEM job schedulers and report managers have become increasingly important over the last few years.

A.2 When are Courses Scheduled and When are they Needed?

It is best to schedule training close to the time the new skill is needed. Some OS/390 training is required early in the migration. An example is the specification phase, when some knowledge about OS/390 is required early in the project, to help define the target system.

It can be a waste to have the application programmers trained on TSO early in the migration. The skill is typically not needed until the system testing phase of the migration.

A.3 Who will Provide the Training?

Hiring a skilled MVS person for the migration, whether temporary or permanent, helps with the skill level of that one person. Don't assume that the person is a skilled educator or will have time to teach a curriculum to other team members in his/her spare time. It is likely that this person will have a list of responsibilities that may not include being a trainer and may not know VSE at all.

Benefits of on-site training

- Curricula can be customized to suit your specific needs
- Blocks of training (for example, one week) can be tailored to deliver the training when it is needed for the migration

IBM Education and Training (E&T) can help you with your training challenge by providing customized training. E&T specializes in presenting a common basic curriculum to a mixed audience of operators, applications programmers and system programmers (OS/390 basics, how to use ISPF, JCL basics) and then splitting each audience to learn those topics specific to their job performance.

For more information and for assistance in designing a custom roadmap of training, contact us at www.training.ibm.com/ibmedu/custom.

A.4 Where will the Training Take Place?

There are a number of items to consider regarding where training takes place. Among them are cost, how the time the class is offered fits with the project needs and the availability of the students.

One consideration is that if you choose to bring in a trainer for a particular class, consider delivering the class somewhere other than the normal work location such as across town at a hotel. This can provide just enough distance to keep the students from being drawn into daily problems at the office.

Having training delivered on site versus go away schools provides significant advantage in class scheduling. The on-site class can be scheduled for the time the skill is ready to be used. For go away classes you are normally limited to the specific times the class is offered.

Appendix B. Mapping ISV Products and Functions

This is a frequent topic of discussion with customers considering migrating. How a customer's ISV products map to those contained in base OS/390 is always discussed.

It is a common migration task to ensure there is equality. This equivalent function mapping will grow in importance as coexistence is established.

It is also a way customers can reduce their ISV SW stack charges. Also if customers are not familiar with the comparable functions of each operating system it provides a good guide and helps when it is time to configure the OS/390 system.

This mapping information also fits in with the description of system management products listed in 2.2, "OS/390 Components/Products/Subsystems" on page 18.

B.1 The IBM Software Migration Project Office (SMPO)

The SMPO is part of IBM's North America Software Group and has been in business since 1993 providing solutions that help solve customers' business problems. We do that by helping companies, like yours, migrate to the industry's leading systems management and database: DB2 and IBM/Tivoli products.

Many of our customers have made substantial investments in platforms which no longer support their growing business requirements. They have turned to us for help because the SMPO has the experience, the skills and the tools to assist them in executing a successful migration project.

IBM/Tivoli product offerings include a full suite of S/390 system and network management products for: system automation, enterprise job scheduling, security management, storage management, performance management and report management. These products and service offerings can provide alternatives to ISV products.

If you have any questions or need more information about the SMPO, feel free to navigate about our home page, or you can contact any one of our IBM/Tivoli Migration Team or Database Team personnel. The SMPO home page can be found at:

- http://www.ibm.com/Solutions/softwaremigration/

B.2 VSE ISV System Management Products and OS/390 Compared

Table 46 (Page 1 of 3). S/390 Software Product Mapping

Vendor	Vendor Product	IBM Product	PID #	Primary Function(s)	Migration Services
BMC	DB2 Recovery Plus	DB2	5695-DB2	DB utility stand func of DB2; perf consideration	
BMC	DB2 Reorg Plus	DB2	5695-DB2	DB utility	
BMC	DB2 Unload Plus	DB2	5695-DB2	DB utility	
BMC	Delta IMS-DB/DD	IMS Utilities	5685-093	IMS utility	IGS
BMC	Fast Reorg Plus	IMS Utilities	5685-093	IMS utility	IGS
BMC	Image Copy Plus	IMS Utilities	5685-093	IMS utility	IGS
BMC	Load Plus	IMS Utilities	5685-093	IMS utility	IGS

Table 46 (Page 2 of 3). S/390 Software Product Mapping

Vendor	Vendor Product	IBM Product	PID #	Primary Function(s)	Migration Services
BMC	Unload Plus	IMS Utilities	5685-093	IMS utility	IGS
CA	CA-ADS/Online	CSP/AD CSP/AE		High Level Language (IDMS)	
CA	CA-Alert/CICS	RACF	5645-001	CICS Security	Tivoli Migration Team
CA	CA-ASM2	DFSMS/hsm	5645-001	Host backup, archival for DASD	Tivoli Migration Team
CA	CA-DATACOM/DB	DB2	5695-DB2	Database	SMPO/DB
CA	CA-Driver	OPC	5697-OPC	Job scheduling	Tivoli Migration Team
CA	CA-Dynam/TMLS	DFSMS/rmm	5645-001	Tape management	Tivoli Migration Team
CA	CA-EARL	PR/MVS	5695-101	Report writer	Tivoli Migration Team
CA	CA-EASYTRIEVE	QMF	5706-254	SQL queries and reports	SMPO/DB
CA	CA-EASYTRIEVE +	QMF	5706-254	SQL Queries	SMPO/DB
CA	CA-Explore	RMF	5645-001	Performance monitor	Tivoli Migration Team
CA	CA-Express Delivery	RMDS	5648-048	Report distribution	Electronic Archive Solutions
CA	CA-Extended/DASD			Compress VSAM clusters	
CA	CA-FAQS/XP	Netview6000	5696-731	Console, message management	IGS
CA	CA-FAVER			Backup and restore performance enhancer	
CA	CA-HYPER-BUF			VSAM buffer manager	
CA	CA-IDEAL	CSP/AD+CSP/AE		4G Appl Development System	
CA	CA-IDMS-Culprit	DB2 or IMS/DB		Database	SMPO/DB
CA	CA-IDMS-UCF	None		no IBM product needed; CICS inherently supports	
CA	CA-IDMS/DC	IMS, CICS, DB2		Database and transaction monitor	SMPO/DB
CA	CA-INTERTEST	CICS ExtDiag		Interactive Program Testing	
CA	CA-JARS	PR/MVS	5688-101	Accounting and chargeback	Tivoli Migration Team
CA	CA-JARS/CICS	PR/MVS	5688-101	CICS performance monitor	Tivoli Migration Team
CA	CA-Librarian	ISPF/SCLM	5645-001	Source Code Control	IGS - Tom Hartrick
CA	CA-LOOK	RMF	5645-001	Performance monitor	Tivoli Migration Team
CA	CA-MasterCAT			Optimizes MVS buffer spaces	
CA	CA-MetaCOBOL+	CSP/AD+CSP/AE		High Level COBOL Development	
CA	CA-Netman	INFO-MGMT	5695-171	Data Center administration	Tivoli Migration Team
CA	CA-Opera	SA/390	5645-005	Automated operations	Tivoli Migration Team
CA	CA-Optimizer	COBOL II	5688-023	COBOL optimizer	IGS
CA	CA-Panvalet	ISPF/SCLM	5645-001	Source Code Control	IGS - Tom Hartrick
CA	CA-Scheduler	OPC	5697-OPC	Job scheduling	Tivoli Migration Team
CA	CA-Sort	DFSORT	5645-001	Host sorting	IGS
CA	CA-SRAM	DFSORT	5645-001	Host sorting	IGS
CA	CA-Top Secret	RACF, OS390 SS	5645-001	Security	Tivoli Migration Team
CA	CA-Verify	CICS (MVS)	5655-147	CICS quality assurance	IGS
CA	CA-VSAMAID			VSAM cluster stats & tuning	
Candle	Omegamon II for SMS	DFSMS/optimizer	5695-DF1	DASD performance tool	Tivoli Migration Team
Generic Sys	DR.D	DFSMS/dss	5645-001	Fast dump restore	Tivoli Migration Team
McKinney	CICS/Hotprint			CICS utility	
McKinney	CICS/Message			CICS utility	

Table 46 (Page 3 of 3). S/390 Software Product Mapping

Vendor	Vendor Product	IBM Product	PID #	Primary Function(s)	Migration Services
Mobius	Infopac report	RMDS	5648-048	Report distribution	Electronic Archive Solutions
Mobius	Infpopac Schedule	OPC	5697-OPC	Job scheduling	Tivoli Migration Team
Oracle	Oracle	DB2	5695-DB2	Relational database	SMPO/DB
Soft AG	ADABAS	DB2	5695-DB2	Database	SMPO/DB
Syncsort	Syncsort	DFSORT	5645-001	Host sorting	IGS

Note: PID #s and Product Functions should be checked by the user prior to ordering any software.

Appendix C. DFSMS Naming Conventions

This chapter was written by John Tyrrell of IBM′s Storage Systems Division. John is one of the original senior architects of DFSMS. He also invented TMM (Tape Management Methodology) and is the author of the Volume Mount Analyzer program which is part of DFSMSdfp. He is the senior architect/inventor of the DFSMS Optimizer product, one of the major components of DFSMS. John interacts with a wide variety of IBM customers, and has personally been to over 600 customer data centers and participated in over 350 tape and DASD studies to better understand and address IBM customer needs.

John spent 10 years in application development of a system with six million lines of code which ran in over 60 IBM data centers. Many of the data set naming techniques in this document were developed based on both John′s application experience as well as working with IBM customers.

This chapter offers suggestions on the naming of data sets such that you would be able to fully exploit the technology of both DFSMS and MVS/ESA. It does not imply that you must rename all of your data, nor does it imply that these are the only conventions that will work under DFSMS and MVS/ESA.

These naming conventions are suggestions resulting from many visits and interactions with IBM internal and IBM customer application areas. These suggestions reflect both the opinion and the experience of the author, and as such, do not necessarily represent the view of the IBM Corporation.

As with all standards, they should act as a guide for the reader. The reader, in this case, should really be the application designer. This is the person who will set up all of the JCL, CLISTs, ISPF panels, JCL skeletons, JOB networks and so on in order to run the major application. He or she is usually the person who sets up the standards to be applied to all procedures involved with running a major application.

It is recommended that the reader go through the entire chapter, and then build his or her own set of rules based on the suggestions offered here as applied to the particular application in question.

C.1 Data Set Naming Guidelines

The purpose of this chapter is to offer some guidelines as to what constitutes proper data set naming conventions. This would allow customers to easily exploit the functions of DFSMS for the proper assignment of System Managed Storage (SMS) policies to manage the data. Although the ACS (Automatic Class Selection) Routines will allow the Storage Administrator to filter on more than just data set name, the name of the data set is fundamentally important. Some of these advantages are listed below:

- It allows more flexibility for the assignment of levels of service to data sets
- It is easier to write/maintain ACS Routines
- The ACS Routines are more durable (that is, meaningful over time)
- Data set filters can be useful for other storage management techniques (for example, ISMF, DFDSS)

There are two basic pieces of information that one should be able to obtain from the data set name:

1. Who owns it?
2. What is it?

The following section will highlight all of the basic components that could potentially be used in a data set naming standard.

C.2 Components of a Data Set Name

Not all of the following levels of qualification are necessary for naming data sets. Instead, these represent some common levels of qualification that one tends to find in a good and meaningful data set naming convention. Some of the qualifications make sense for certain types of data while other levels don't. This list is intended to be a superset of all possible types of qualification levels.

Also, not all of these levels have to be coded as a separate level of qualification (that is, separated by periods). Other possibilities are via positional characters within a given qualifier. The one exception to this is the High-Level Qualifier (HLQ). You should not create unnecessary numbers of these due to positional characters. This is explained in more detail in the next section.

C.2.1 High-Level Qualifier (HLQ)

The HLQ should identify who "owns" the data. This could be for things such as billing purposes, or simply for locating the owner in case of a problem. It may represent a user ID, a project, an application, a business unit or a group. It may also represent a sharing of the same set of data by a set of individuals from a security standpoint (for example, as the notion of RACF **userid** and **groupid**).

There should be no other levels of qualification imbedded in this portion that would tend to artificially multiply the number of HLQs in an installation. The goal should always be to minimize the number of HLQs to the point that they serve the management purpose (that is, billing, identification and so on).

It may be important to even have a standard convention within the HLQ. For example, all TSO user IDs begin with a "$" as the first character -- this would allow the Storage Administrator to easily avoid filter collisions in ACS Routines. Again, the goal of doing this would be to allow the Storage Administrator to do his job more easily in that all TSO data could be simply filtered out. The trade-off here, of course, would be with the usability of the TSO LOGON IDs.

Another trade-off would be the intent of managing groups of application data. It may be more important, for example, to associate certain TSO LOGONs with the particular application area so that large applications could easily be identified and moved by filtering on the first character of the HLQ. There is also a usability problem here in that the TSO user would have to keep changing his ID if his job changed from one large application to another. This also means that electronic mail might be a problem if individual users have a lot of LOGON ID changes -- this also has some security implications as well (electronic mail being sent to the wrong ID).

Note: As a personal recommendation from the author, it has been found that one tends to cause more problems by choosing user IDs that will definitely change due to such things as career changes. A better way is to have

individuals keep a standard LOGON ID, and change the set of filters instead of the IDs.

As an example of this set of conventions, consider a common problem of constantly shifting workloads. If the Storage Administrator was always faced with getting the data for a given application and moving it to another system, then it would make good sense to have a naming convention for the HLQ that would allow him to easily accomplish this via filtering techniques. Other reasons are for data portability to other installations for disaster/recovery situations that would cause an application to be brought up on an alien system.

One example of a naming convention for HLQs might be the following:

- First Character:
  - A - Accounting Support
  - D - Documentation
  - E - Engineering
  - F - Field Support
  - M - Marketing Support
  - P - Programming
  - $ - TSO user ID

    An alternative here is to use one of the characters above as the first character of the TSO user ID to allow movement of all data within an application (including the TSO data). This is **not** the way recommended by the author -- the preference would be a standard code for TSO, such as ″$″.

    The above list does **not** represent **all** of the possibilities. For example, a bank might have separation of HLQs by major application such as checking, savings, mortgage loans, investments or ATM. An insurance company might have applications such as life, auto, personal insurance, major medical and corporate accounts.
  - Remaining Characters = Project Name or Code Number

    Note: This might have been chosen because a project code might exist for programming, engineering, documentation, and accounting, but it does **not** imply that one **must** do it that way. It has been the experience of the author that similar project codes would not be common.

  A more natural idea is to have the actual project name following the first character. For example, the remaining portion of the ″E″ project could be the code name of the machine being designed, while a ″P″ project could be the name of the programming application or product name.

  Some examples might be:

  - E3090M150
  - E3090M200
  - E3090M300
  - E3090M400
  - E3090M500
  - E3090M600
  - E3380K
  - E3380J

- E3380D
- E3380S
- E3990M3
- E3990M2

The above example would allow various filtering techniques the flexibility of recognizing different sets of data easily. Some examples of this follow:

- HLQ = E* -- All of the Engineering Data
- HLQ = E3090* -- All of the 3090 CPU Family of Designs
- HLQ = E3380* -- All of the 3380 DASD Family of Designs

C.2.2 Relative Importance

This level of qualification might indicate things such as:

- Production Data
- Development Data
- Test Data

In general, it would be important to be able to recognize the distinction between production data and test data. Other types of levels could be:

- Master Data
- Update Data
- Work Data

C.2.3 File Contents

This level of qualification should state what the data is. For example, an application strips some information out of the master database and builds a work file for subsequent processing. This file contains the employee id number and his job code. One might then call this file the ″employee- job code file″. Other examples might include such things as:

- Telephone call log (TELPHLOG)
- Parts inventory file (PRTINVEN)
- Parts unit cost file (PRTUCOST)
- Payroll file (PAYROLL)
- Checking account transaction file (CHKXACTN)
- CADAM circuit design file (CKTDESGN)
- Heat dissipation statistics file (HEATSTAT)
- Simulation result file (SIMRSLTS)
- Program source (PGMSRCE)
- Life insurance account file (LIFEACCT)
- User′s Manual script file (USMANUAL)
- Input manufacturing file (INPMANUF)
- Transportation bill of lading file (XPRTBILL)

All of the above examples describe what the data is. There should be a unique character code for each data set type within a given application. This concept is demonstrated with each of the examples above.

Note: Eight characters have been used as the standard for the data set type in all of the above examples. This is probably a reasonable number of characters, although not mandatory. It may not even be a good idea to make file names too

readable, such as, a file called **MASTER.PAYROLL.WEEKLY**. This might be just too tempting to your average system hacker. A better choice might be **MR.PY.WK.**

C.2.4 User Name

This qualifier should allow the end creator to assign their own unique name to identify the particular set of a certain type of data; for example, with a master circuit file, there would be a distinction between MYPART and YOURPART, or MYPART1 versus MYPART2. Some other examples of these are:

- Part number (#0135678)
- Print program (PRTPROGM)
- New York Area (NEWYORK)
- X15 Model (X15MODEL)
- Geological site #458 (GEO#458)
- Branch office #57 (BROFC057)

The intent of this level of qualification is to uniquely identify one piece of a type of data from another piece of the same type of data. Examples of this show the distinction between ″TELPHLOG.NEWYORK″ and ″TELPHLOG.CALIF″ or between ″PGMSRCE.MYPGM1″ and ″PGMSRCE.MYPGM2″. This should use the entire qualifier (that is, all of the eight characters).

C.2.5 Data Set Level

Another level of qualification that is sometimes useful in application areas is the level of the piece of data. Some examples of this are listed below:

- Design level or version or release level (for example, for engineering, programming, documentation)
- Change number, an arbitrary number to indicate a constantly increasing number for subsequent improvements on a piece of data
- Cyclic level (for example, yearly, monthly, weekly, daily)

C.3 Things Not to Include in the Data Set Name

There are certain pieces of information that should never be part of the data set name. The general category of this data is that information which is very likely subject to change. This type of qualification usually doesn′t add anything meaningful in terms of identifying the data for storage management reasons. Some examples of this are shown in subsequent sections.

C.3.1 Department Number

This is a piece of information that is sure to change either due to re-organization, or movement of projects or individuals.

C.3.2 Application Location

If this application ever got moved to another site, then all of the data sets would have to be renamed. In some cases, it could be important to name the data by a location name, for example, CHICAGO. This could be perfectly acceptable in certain cases. Suppose there were two telephone directory files, ″TELPHDIR.CHICAGO″ and ″TELPHDIR.PODUNK″. This would be okay. That′s because it is very unlikely that either Chicago or Podunk would ever move.

Application workload is another story. For example, if an application currently runs in Chicago, or Podunk, one might be tempted to include this as part of the data set name for disaster/recovery purposes. If there is a chance of moving these applications to run in a different location, then it would be a bad idea to imbed it into a name.

C.3.3 Management Criteria

This type of generic criteria that some people recommend is the notion of identifying disaster/recovery data, or vital records, for example within the data set name. In general, it is not a good idea to imbed management criteria within the data set name, since it is mainly driven by the technology at hand. They may also change for business reasons (for example, a change in the state laws).

If either the technology or the business reason changes, then the data set would have to be renamed to match it. One should simply name the data for what it is and keep its management policy separate.

Therefore, it is **not** a good idea to specify qualifiers such as ″DR″ or ″VITALREC″. By always naming the data for what it is, the policy for managing it can be kept separately without ever needing to rename the data.

C.3.4 Output Device Type

This is a general class of information that is also based on a certain level of technology. As technology improves, you may very well decide to change the way it is managed. For example, you might put a data set qualifier of MICROFIC to indicate that this data set should be put out to microfiche -- whereas somewhere in the future, one could envision this same piece of data being put to a high-speed link which is connected to some automated high-capacity storage device.

Qualifiers such as ″TAPE″ or ″DISK″ or ″T3480″ or ″T3490″ should never be coded in data set names. This type of qualifier is destined for change as newer device technologies are created.

C.3.5 Expiration Date

The **EXPDT** and **RETPD** allocation keywords associated with the data set are another form of management criteria in that they specify the purge date for data. These keywords clearly put storage management in the hands of the end users.

To change the policy, you must change the date associated with the data set. This is just as bad as forcing the application to go back and re-figure the new management criteria. It represents a policy of sorts and therefore, should be separated from the name itself. This type of information should be allowed to change without actually changing the name of the data.

C.3.6 Access Method

At one point in time, many installations adopted a policy of distinguishing VSAM data from non-VSAM data. The reality behind this standard was that there were many functions that were not supported for VSAM and this was an easy way to recognize that data.

The main reason for this distinction was because of old VSAM catalogs and the ″ownership″ of the volume and data by VSAM. In order to avoid these problems, you had to separate the VSAM data from the non-VSAM data by catalog. We have come a long way since then and this notion is no longer needed with the use of ICF catalogs. There should be ***no*** distinction because of access method.

C.3.7 Job Name

Some installations have used the name of the job which created this data set. This is certainly a piece of information which is very likely to change. It usually says very little about what this piece of data actually is, since the job usually creates all sorts of data set types. It is not a good practice to include this information as a part of the data set name.

C.4 Common Applications - Naming Conventions

This section will focus on some of the common MVS applications that tend to put out a fair amount of data on the system and the suggestions for an associated naming convention.

C.4.1 TSO Naming Conventions

TSO certainly has a very recognizable data set naming convention. The rules are fairly simple and easy to understand:

- Three levels of qualification: **userid.dsname.dstype**
- Standard set of data set types (for example, CLIST, FORT, PLI, CNTL, and so on)

All of the TSO functions, commands and also the ISPF/PDF functions tend to complement this naming convention. Therefore, for ease of use and transportability, it is generally a good idea to maintain this convention.

Some applications run with production-type data under TSO using the standard JCL PROCs, CLISTs, and PANELS that would be used for the normal production data. For example, consider a programming application that had a naming convention of:

```
library.dstype.project.version.release
```

where:

```
library:  (PROD, DEVL, or TEST)
dstype:  (SOURCE, MACRO, LOAD, OBJ, JCL, and so on)
project:  (APPLIC1, APPLIC2, . . .)
version:  (V1, V2, . . .)
release:  (R1, R2, . . .)
```

It would certainly be acceptable for unit testing on certain data to be done under a TSO user ID such that the user ID would be substituted for the ″library″ and all of the remaining qualifiers would stay the same. The key point here is the

usability of the system and the need to manage it differently based on what set of data this actually is.

C.4.2 VSAM Data Set Naming Conventions

Many companies have decided that it is a good idea to associate all of the VSAM components with the base cluster by some recognition pattern. Some of the reasons behind this have to do with billing, and also with the usability of doing catalog locates and so on, to find all of the associated components easily with available software technology.

The normal standard that has been adopted in most cases is the first portion of the name being the cluster name and the component name of DATA, INDEX, or AIX name. For example, consider a VSAM cluster of X.Y.Z that was a KSDS with two AIXs which were also keyed data sets. Also, assume that there were two path names defined over the alternate indexes. The set of names would be:

X.Y.Z
X.Y.Z.DATA
X.Y.Z.INDEX
X.Y.Z.PATH1
X.Y.Z.AIX1
X.Y.Z.AIX1.DATA
X.Y.Z.AIX1.INDEX
X.Y.Z.PATH2
X.Y.Z.AIX2
X.Y.Z.AIX2.DATA
X.Y.Z.AIX2.INDEX

C.4.3 DB2 Naming Conventions

The standard data set naming convention for all DB2 data sets is the following:

```
hlq.DSNDBx.dbname.tblspacename.I0001.A00n
```

where:

DSNDBx is

- *DSNDBC* -- cluster
- *DSNDBD* -- data component
- *DSNDBI* -- index component

dbname -- database name (user selected)

tblspacename -- table space name (user selected)

I0001 -- hard-coded constant

A00n -- where ″n″ is system generated for extensions to the table space

Although the DB2 naming convention is certainly distinguishable, it is difficult to associate different management policies for different sets of data within a particular database. For example, in a programming application, one can easily imagine different management policies for things such as SOURCE, MACRO, SCRIPT and so on, versus things such as OUTLIST, LISTING, TESTLIST, LIST, ASMLIST and so on.

Not all data within an application has the same management criteria --in the case of DB2 applications, it is difficult to place distinguishable characteristics

within the data set name so that ACS Routines can easily ascertain one piece of data from another. The only alternative to this is to place a very restrictive convention on the qualifiers that can be specified.

For example, you could break down the table space name by having positional characters represent different levels of qualification for the data. The following is one possibility:

- first character: P -- production, T -- test
- second character: R -- report, I -- intermediate file, M -- master DB
- third character: W -- weekly, D -- daily, M -- monthly
- fourth-eighth characters: table space name

Until this restrictive naming convention is lifted, the only other alternative is to use the available free characters to distinguish the data within the database application.

C.4.4 Generation Data Sets

The notion of a GDG (Generation Data Group) is that for a given name, say ″A.B.C″ there could be many generations. Each generation data set (GDS) would have the following name form:

A.B.C.GnnnnV00

where:

nnnn = next number in sequence (may wrap around)

The only naming convention difference here is that the application loses one level of qualification, mainly the lowest level. Other than that, there is no real distinction between GDSs and other data set names. It is generally not a good idea, however, to use the generation to indicate a different type of data; for example, the idea of the ″+1″ version being a report, the ″+2″ version being an intermediate file. The first portion of the name should state what the data is and the generation should represent a later level or generation of that data. A better idea would be to have a separate GDG for reports, intermediate files and so on.

Appendix D. Special Notices

This publication is intended to help customers and IBM technical personnel to migrate from VSE to OS/390.

The information in this publication is not intended as the specification of any programming interfaces that are provided by VSE or OS/390. See the PUBLICATIONS section of the IBM Programming Announcement for VSE/ESA and OS/390 for more information about what publications are considered to be product documentation.

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

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

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

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

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of

including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

3090
ACF/VTAM
AD/Cycle
ADSTAR
Advanced Function Presentation
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
APPN
AS/400
BookManager
BookMaster
C/370
CallPath
CBIPO
CBPDO
CICS
CICS/ESA
CICS/MVS
CICS/VSE
CICSPlex
COBOL/370
Common User Access
CT
CUA
Current
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DFSMS
DFSMS/MVS
DFSMSdfp
DFSMSdss
DFSMShsm
DFSMSrmm
DFSORT
Distributed Relational Database Architecture
DRDA
ECKD
eNetwork
ES/9000
ESA/370
ESA/390
ESCON
FFST
FunctionPac
GDDM
Hardware Configuration Definition
Hiperbatch
IBM
ImagePlus
IMS
IMS/ESA
InfoPrint
Intelligent Printer Data Stream
IP PrintWay
IPDS
MO:DCA
MQSeries
Multiprise
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
Net.Data
NetSpool
NetView
OfficeVision
OPC
Open Class
OpenEdition
Operating System/2
OS/2
OS/390
OS/400
Parallel Sysplex
PR/SM
Print Services Facility
Processor Resource/Systems Manager
ProductPac
PSF
PSF/6000
QMF
RACF
RAMAC
Resource Measurement Facility
RETAIN
RMF
RS/6000
S/370
S/390
SAA
ServicePac
SOMobjects
SP
SQL Master
SQL/DS

SupportPac
System/360
System/370
System/390
SystemPac
Systems Application Architecture
SystemView
Virtual Machine/Enterprise Systems Architecture
VisualAge
VisualLift
VM/ESA
VM/XA
VSE/ESA
VTAM

The following terms are trademarks of other companies:

1-2-3 is a trademark of Lotus Development Corporation

ADE is a trademark of Loral/Rolm Mil-Spec

ATM is a trademark of Adobe Systems, Incorporated

C-bus is a trademark of Corollary, Inc.

C++ is a trademark of American Telephone and Telegraph Company, Incorporated

CA is a trademark of Computer Associates

CADAM is a trademark of Cadam Incorporated

CORTEX-MS is a registered trademark of SISRO Inc

CSS is a trademark of CSS Laboratories, Incorporated

DCE is a trademark of The Open Software Foundation

Domino is a trademark of Lotus Development Corporation

Encina is a trademark of Transarc Corporation

Express is a trademark of Parasoft Corporation

HP/UX is a trademark of Hewlett-Packard Company

Intel is a trademark of Intel Corporation

Lotus is a trademark of Lotus Development Corporation

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

MS is a trademark of Microsoft Corporation

Network File System is a trademark of Sun Microsystems, Incorporated

NFS is a trademark of Sun Microsystems Incorporated

OMEGAMON is a trademark of Candle Corporation

ONC is a trademark of Sun Microsystems, Incorporated

Open Software Foundation is a trademark of The Open Software Foundation, Incorporated

Oracle is a trademark of Oracle Corporation

OSF is a trademark of Open Software Foundation, Incorporated

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

PKZIP is a trademark of PKWARE, Incorporated

POSIX is a trademark of Institute of Electrical and Electronic Engineers

PostScript is a trademark of Adobe Systems, Incorporated

Report Manager is a trademark of Image Products, Incorporated

SMS is a trademark of Standard Microsystems Corporation

Solaris is a trademark of Sun Microsystems, Incorporated

Sun Microsystems is a trademark of Sun Microsystems, Incorporated

Tivoli, TME, and TME 10 are trademarks of Tivoli

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

X/Open is a trademark of X/Open Company Limited

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

Appendix E. Related Publications

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

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 561.

E.1.1 OS/390 and MVS Redbooks

Book Title	Publication Number
ESCON MVS Operator Problem Determination	GG66-3239
OS/390 Software Management Cookbook	SG24-4775
Parallel Sysplex Configuration Cookbook	SG24-4706
MVS Multisystem Consoles in a Sysplex	SG24-4626
Planning for CA-ACF2 Migration to OS/390 Security Server	SG24-4663
MVS 3.1.3 and RACF 1.9 Security Implementation Guide	GG24-3585
RACF V2.2 Installation and Implementation Guide	SG24-4580
OS/390 Parallel Sysplex Capacity Planning	SG24-4680
JES3 in a Parallel Sysplex	SG24-4776

E.1.2 Other Redbooks

Book Title	Publication Number
IBM Network Products Implementation Guide	GG24-3649
R/390 (P/390) New User Cookbook *	SG24-4757
Automation for S/390 Parallel Sysplex	SG24-4549
ES/9000 Operating Your System, Volume 1	SA24-4350
ES/9000 Operating Your System, Volume 2	SA24-4351
9221 Cookbook	GG24-3935
HCD Primer	SG24-4037
ESA/390 Microprocessor (C, E, P, and R1 Models)	GG24-4497
Continuous Availability with PTS	SG24-4503
ESA/390 Microprocessor (R2 and R3 Models)	SG24-4575
ESCON Implementation Guide	SG24-4662
OSA-2 Implementation Guide	SG24-4770
HMC with S/390 CMOS Processors	SG24-4832
S/390 G3 Enterprise Server: CSAR Presentation Guide	SG24-4911
Interoperability between VSE DL/I and OS/390 IMS DBCTL	SG24-5249
PSF/VSE Application Programming Guide	S544-3666
AFP Printing in a Cross-System Environment	GG24-3765

E.2 OS/390 Product Publications

See GC28-1727, *OS/390 Information Roadmap* for a complete list of OS/390 books.

E.2.1 Planning Books

The following hard-copy books are part of the *OS/390 Installation Planning Kit* which can be ordered by publication number GK2T-6710:

Book Title	Publication Number
OS/390 Introduction and Release Guide	GC28-1725
OS/390 Information Roadmap	GC28-1727
OS/390 Planning for Installation	GC28-1726
SystemView for MVS Up and Running!	GC28-1241
The Year 2000 and 2-Digit Dates: Guide	GC28-1251
OS/390 BookManager Hints and Tips	GC28-1987

Other OS/390 Books: There are many other introductory and planning books available in the OS/390 library. Here are some of the most important ones you should be familiar with:

Book Title	Publication Number
Custom-Built Offerings Planning	SC23-0352
ServerPac: Using the Installation Dialog	SC28-1244
OS/390 MVS Planning: Operations	GC28-1760
OS/390 MVS Planning: Global Resource Serialization	GC28-1759
OS/390 MVS System Data Set Definition	GC28-1782
OS/390 MVS Device Validation Support	GC28-1748
DFSMS/MVS General Information	GC26-4900
DFSMS/MVS Library Guide	GC26-4902
DFSMS/MVS Planning for Installation	SC26-4919
OS/390 TSO/E General Information	GC28-1964
OS/390 JES2 Introduction	GC28-1794
IBM BookManager READ/MVS and BUILD/MVS: General Information	GC38-2032
IBM BookManager READ/MVS: Getting Started	SC38-2033
OS/390 Printing Softcopy BOOKs	S544-5354
OS/390 HCD Planning	GC28-1750
OS/390 HCM User's Guide	SC33-6595
OSA Planning	GC23-3870
MVS Planning: Security	GC28-1439
OS/390 Security Server (RACF) Introduction	GC28-1912
OS/390 ISPF Getting Started	SC28-1294
OS/390 SDSF Guide and Reference	SC28-1622
HLASM General Information	GC26-4943
GDDM General Information	GC33-0866
OS/390 Parallel Sysplex Overview	GC28-1860
OS/390 MVS Planning: Workload Management	GC28-1761
CustomPac Installation Dialogs	SA22-7240
OS/390 MVS Recovery and Reconfiguration Guide	GC28-1777

E.2.2 OS/390 Online Product Library

Book Title	Publication Number
OS/390 Online Collection	SK2T-6700

E.3 Other Publications

Book Title	Publication Number
RACF General Information	GC23-3723
Getting Started with DFSORT	SC26-4109
Get DFSMS FIT: Fast Implementation Techniques	SG24-2568
DFSMS FIT: Fast Implementation Techniques Process Guide	SG24-4478
DFSMS FIT: Fast Implementation Techniques Installation Examples	SG24-2569

E.4 Other Sources

E.4.1 Books on the Internet

There are several IBM world-wide web sites available with more information:

E.4.1.1 Redbooks

See ***http://www.redbooks.ibm.com/***. for OS/390 Publications on the Internet.

E.4.1.2 OS/390 Books

See ***http://www.s390.ibm.com/os390*** and click on ″THE LIBRARY″.

E.4.1.3 IBM Printing Systems

See the IBM Printing Systems Company Web Site at ***http://www.printers.ibm.com/***.

E.5 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com/.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/

- **PUBORDER** — to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.html). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	Internet
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Numerics

2-digit-year format. A format that provides a year date as two digits only to represent a year within a specific century. The two high-order digits of the year are truncated; for example 1995 is represented as 95.

4-digit-year format. A format that provides a year date as four digits: the two high-order digits represent the century and the two low-order digits represent the year within the century. For example, 1995 represents the year 1995; 2095 represents the year 2095.

20th century. The period of time 0000.00 hrs 1901-January-1 through 2400.00 hrs 2000-December-31.

21st century. The period of time 0000.00 hrs 2001-January-1 through 2400.00 hrs 2100-December-31.

24-hour clock. A clock that keeps time from 0000 (midnight) to 1200 (noon) and from 1200 (noon) to 2400 (midnight). Compare with 12-hour clock.

3270 emulation. The use of a program that allows a device or system such as a personal computer to operate in conjunction with a host system as if it were a 3270- series display station or control unit.

A

abend code. A system code that identifies the system message number and type of error condition causing the abend.

abnormal termination. (1) The cessation of processing prior to planned termination. (2) A system failure or operator action that causes a job to end unsuccessfully.

access control. In computer security, ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

access level. In computer security, the level of authority a subject has when using a protected resource; for example, authority to access a particular security level of information.

access method. A technique to obtain the use of data, storage, or the use of an input/output channel to transfer data; for example, random access method, sequential access method.

access method routines. Routines that move data between main storage and input/output devices.

Access Method Services (AMS). A utility program (named IDCAMS) that defines VSAM data sets (or files), allocates space for them, modifies attributes, and manipulates data sets and catalog entries.

access mode. The way a file is used within a job step, a program, or a module. Most access modes correspond to OPEN modes specified in OPEN statements (such as input, output, or update).

access path. A sequence of data items used by a database management system to access records or other data items stored in a database. There may simultaneously exist more than one access path for one data item.

account file. A direct access file maintained by VSE/POWER to hold the accounting information it generates and the programs that it controls.

actual conversion. The conversion of source material done at the end of migration, in order to switch over from VSE to MVS. (Contrast with dummy mass conversion.)

AD/Cycle. An IBM product that offers an enterprise modeling approach supported by tools that will assist in the creation of an enterprise model to be validated, analyzed, and then used to generate applications. It consists of a framework for, and a set of, application development tools provided by an Application Development (AD) platform, designed to support the integration of tools through a consistent user interface, workstation services, an AD information model, tool services, Repository Services, and Library Services. It provides control for defining and sharing application development data.

address space. (1) The range of addresses available to a computer program. (2) The complete range of addresses that are available to a programmer. See also virtual address space. (3) In VSE, a subdivision of the total of virtual storage if the computer system operates in 370 mode.

address translation. In virtual storage systems, the process of changing the address of an item of data or an instruction from its virtual storage address to its real storage address.

AFP. Advanced Function Presentation. A set of licensed programs, together with user applications, that use the all-points-addressable concept to print on presentation devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information.

AFPDS. AFP data stream. A presentation data stream that is processed in AFP environments.

MO:DCA-P is the strategic AFP interchange data stream, and IPDS is the strategic AFP printer data stream.

alphabetic character. Any one of the letters A through Z (uppercase and lowercase). Some licensed programs include as alphabet characters the special characters #, $, and @.

alphanumeric. Pertaining to data that consist of letters, digits, and usually other characters, such as punctuation marks.

alternate COPY. A source library management feature that provides text inclusion within source modules, JCL streams or any other card-image data, in contrast to the standard source text inclusion features that various compilers provide.

alternate index. In systems with VSAM, a collection of index entries related to a given base cluster and organized by an alternate key, that is, a key other than the prime key of the associated base cluster data records; it gives an alternate directory for finding records in the data component of a base cluster.

AMS. See Access Method Services.

analysis routine. A routine that analyzes error records, provided by an error handler, to isolate failures to one or more field replaceable units (FRUs).

APA. all points addressable. The ability to address, reference, and position text, overlays, and images at any defined position or pel on the printable area of the paper. This capability depends on the ability of the hardware to address and to display each picture element.

APF. authorized program facility. The authorized program facility (APF) is a facility that an installation manager uses to protect the system. In MVS, certain system functions, such as all or part of some SVCs, are sensitive; their use must be restricted to users who are authorized. An authorized program is one that executes in supervisor state, or with APF authorization.

application. A set of programs, JCL jobstreams, and other programming elements written for or by users for their own data processing production.

application program. (1) A program that is specific to the solution of an application problem. Synonymous with application software. (2) A program written for or by a user that applies to the user's work, such as a program that does inventory control or payroll. (3) In SDF/CICS, the program using the physical maps and symbolic description maps generated from a source map set.

application program interface (API). A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

application software. (1) Software that is specific to the solution of an application problem. (2) Software coded by or for an end user that performs a service or relates to the user's work. See also system software.

application step. A job step that executes a user program directly related to data processing production. Contrast with utility step.

archive. A copy of one or more files or a copy of a database that is saved for future reference or for recovery purposes in case the original data is damaged or lost.

array. In programming languages, an aggregate that consists of data objects, with identical attributes, each of which may be uniquely referenced by subscripting.

ascending sequence. The arrangement of data in order from the lowest value to the highest value, according to the rules for comparing data. Contrast with descending sequence.

assembler language. A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

asynchronous processing. A series of operations performed separately from job in which they were requested; for example, submitting a batch job from an interactive job at a workstation. Contrast with synchronous processing.

audit. To review and examine the activities of a data processing system mainly to test the adequacy and effectiveness of procedures for data security and data accuracy. See computer-system audit. See also audit review file, audit trail.

authorization. (1) In computer security, the right granted to a user to communicate with or make use of a computer system. (2) The process of granting a user either complete or restricted access to an object, resource, or function.

authorized library. A library that may contain authorized programs.

authorized program. A system program or user program that is allowed to use restricted functions.

automatic restart. A restart that takes place during the current run, that is, without resubmitting the job. An automatic restart can occur within a job step or at the beginning of a job step. Contrast with deferred restart.

B

background partition. In VSE, a space of virtual storage in which programs are executed under control of the system. By default, the partition has a processing priority lower than any of the existing foreground partitions.

backout. See file and catalog backout.

backup copy. A copy of information or data that is kept in case the original is changed or destroyed.

base cluster. In systems with VSAM, a key-sequenced or entry-sequenced file over which one or more alternate indexes are built. See also cluster.

base register. (1) A register that holds a base address. (2) A general-purpose register that a programmer chooses to contain a base address.

batch application. In VSE, a set of programs that normally processes data without user interaction; for example, an application to print a company payroll. Such an application uses a device, a data file, or the processor intensively for a longer time than online applications.

batch execution. Execution of programs and data that have been submitted or accumulated as batched input.

batch processing. (1) Loosely, the execution of computer programs serially. (2) Pertaining to the technique of executing a set of computer programs such that each is completed before the next program of the set is started.

BCP. Base Control Program. This refers to the ″heart″ of the OS/390 operating system without the JES, RACF, VTAM and other subsystems.

bind. (1) To relate an identifier to another object in a program; for example, to relate an identifier to a value, an address or another identifier, or to associate formal parameters and actual parameters. (2) To associate a variable with an absolute address, identifier, or virtual address, or with a symbolic address or label in a program.

binder. The DFSMS/MVS program that processes the output of language translators and compilers into an executable program (load module or program object). It replaces the linkage editor and batch loader in the MVS/ESA operating system.

bit string. A string consisting solely of bits.

blocking factor. The number of records in a block. A blocking factor is calculated by dividing the size of the block by the size of the record. Synonymous with grouping factor.

bootstrap. A sequence of instructions whose execution causes additional instructions to be loaded and executed until the complete computer program is in storage.

buffer pool. (1) An area of storage in which all buffers of a program are kept. (2) In ACF/VTAM, a group of buffers having the same size. A buffer pool is established at initialization time in the message control program; the buffers are built in extents chained together.

built-in function. (1) A function that is supplied by a language. (2) In PL/I, a predefined function, such as a commonly used arithmetic function or a function necessary to high-level language compilers; for example, a function for manipulating character strings or converting data. It is automatically called by a built-in function reference.

business partner. Any non-IBM organization, with whom IBM has a written contract defining a complementary marketing relationship, that provides end users with information-handling solutions that use or rely upon an IBM offering.

C

C language. A language used to develop software applications in compact, efficient code that can be run on different types of computers with minimal change.

cache. (1) A special-purpose buffer storage, smaller and faster than main storage, used to hold a copy of instructions and data obtained from main storage and likely to be needed next by the processor. (2) A buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

cancel. To end a task before it is completed.

catalog. A collection of all data set information, like device type and volume serial number, that MVS needs to locate a specific data set. Using the catalog simplifies developing MVS JCL that does not change from one run to the next.

catalog backout. See file and catalog backout.

cataloged procedure. A set of control statements placed in a library and retrievable by name.

category (of programming elements). A main classification of programming elements that groups elements of a single type, such as source modules, copied members, and macros.

CBPDO. Custom-Built Product Delivery Offering. A CBPDO is a tape that has been specially prepared for installing a particular product and the related service requested by the customer. A CBPDO simplifies installing a product and the service for it.

CCYY format. A 4-digit-year format that uses two century digits (CC) to indicate the century and two year digits (YY) to indicate the year within the century. The CC representation is provided as either the actual century digits (for example, 18, 19, or 20) or as an encoded value (for example, as 00 to represent 19, 01 to represent 20 as in, 0095 represents the year 1995 and 0195 represents the year 2095.)

century. Although IBM recognizes that the 21st century begins at 0000 hrs, 2001-January-01, for purposes of this document, we are defining the 20th—21st century boundary to be between 2400 hrs, 1999-December-31 and 0000 hrs, 2000-January-1. This allows a discussion of the 21st century to include all dates with a 20yy format inclusive of the year 2000. Hence, the year 2100 is likewise relegated to the 22nd century.

century byte. The high order byte of a field used to contain the two high order digits of a 4-digit year. (For example, 19 in 1995, 20 in 2000 and 2001).

channel-to-channel (CTC). A method of connecting two computing devices.

character set. (1) An ordered set of unique representations called characters; for example, the 26 letters of the English alphabet, Boolean 0 and 1, the set of symbols in the Morse code, and the 128 ASCII characters. (2) A defined collection of characters. (3) All the valid characters for a programming language or for a computer system.

checkpoint data set. A data set that contains checkpoint records.

CICS. See Customer Information Control System.

CICS region. The CICS area of the computer system in which an application is running.

close. (1) A data manipulation function that ends the connection between a file and a program. Contrast with open. (2) To end the processing of a file.

cluster. In systems with VSAM, a named structure consisting of a group of related components; for example, a data component with its index component.

coexistence. The ability of different types of systems to support a program.

command language. A set of procedural operators with a related syntax, used to indicate the functions to be performed by an operating system. Synonymous with control language.

Common Programming Interface. Definitions of those application development languages and services that have, or are intended to have, implementations on and a high degree of commonality across the SAA environments. One of the three SAA architectural areas.

communication region. In VSE, an area of the supervisor that is set aside for transfer of information within and between programs.

compilation. Translation of a source program into an executable program (an object program).

configuration file. A file that specifies the characteristics of a system or subsystem.

console. A part of a computer used for communication between the operator or maintenance engineer and the computer.

context editing. A method of editing a line without using line numbers. To refer to a particular line, all or part of the contents of that line is specified.

control block. A storage area used by a computer program to hold control information. Synonymous with control area.

control language (CL). The set of all commands with which a user requests functions. Synonym for command language. See job control language.

control program. (1) A computer program designed to schedule and to supervise the execution of programs of a computer system. (2) See VM/370 control program, resident control program, IMS/VS control program, VM/XA Migration Aid control program.

conversational. Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought. See also interactive.

conversion (VSE/MVS). A process that modifies VSE applications and data to meet MVS requirements.

copied member. A source text member that can be included in a flow of source data by means of COPY-like statements (COPY statements in Assembler, COBOL, or RPG II; %INCLUDE statements in PL/I; or any other alternate-COPY statements for nonstandard text inclusion).

cosmetic. Referring to a 2-digit-year date that is viewed by human eyes only, such as a print date on hardcopy output or a date on a selection panel. Because it is neither read nor further processed by a program you might be able to exclude its modification from your Year2000 work effort.

Customer Information Control System (CICS). An IBM-licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

CP command. In VM, a command by which a terminal user controls his virtual machine. The VM/370 control program commands are called CP commands. The CP commands that perform console simulation are called console functions.

cross-domain resource. (1) Deprecated term for other-domain resource. (2) In VTAM programs, synonym for other-domain resource.

Custom-Built Installation Process Offering. A product that simplifies the ordering, installation, and service of MVS system control programs and licensed programs by providing them with current updates and corrections to the software that is already integrated.

customization. The process of designing a data processing installation or network to meet the requirements of particular users.

CustomPac. A series of offerings (that is, SystemPac, FunctionPac, ProductPac, and ServicePac) based on PUT levels combined with RSU levels.

cutover. The transfer of functions of a system to its successor at a given moment.

D

DASD. See direct access storage device.

data base description (DBD). In IMS/VS, the collection of macro parameter statements that describes an IMS/VS data base.

data definition language (DDL). A language for describing data and their relationships in a database. Synonymous with data description language.

data exchange. The use of data by more than one program or system. Data recorded or transmitted in a format is referred to as exchange data.

Data Facility Data Set Services (DFDSS). A backup and restore program product.

Data Facility Product (DFP). A program that isolates applications from storage devices, storage management, and storage device hierarchy management.

Data Facility Storage Management Subsystem. An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

data integrity. The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur.

Data Language One (DL/I). 1. In IMS/VS, the data manipulation language that provides a common high-level interface between a user application and IMS/VS. 2. A data base access language used under VSE and CICS/VS.

data management. (1) In an operating system, the computer programs that provide access to data, perform or monitor storage of data, and control input/output devices. (2) In VSE, a major function of the operating system. It involves organizing, storing, locating, and retrieving data.

data migration. The moving of data from an online device to an offline or low-priority device, as determined by the system or as requested by the user. Contrast with staging.

data portability. The ability to use data sets or files with different operating systems. Volumes whose data sets or files are cataloged in a user catalog can be demounted from storage devices of one system, moved to another system, and mounted on storage devices of that system.

data set. Under MVS, a named collection of related data records that is stored and retrieved by an assigned name. Equivalent to a CMS file.

data set control block (DSCB). A data set label for a data set in direct access storage.

data set name. The term or phrase used to identify a data set. See also qualified name.

data space. In VSAM, a storage area defined in the volume table of contents of a direct-access volume to store files, indexes, and catalogs.

data transfer. The movement, or copying, of data from one location and the storage of the data at another location.

database management system (DBMS). A computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The software for using a database may be part of the database management system or may be a stand-alone database system.

DBD. See data base description.

DD statement. Data definition statement.

DDNAME. data definition name. The logical name of a file within an application. The DDNAME provides the means for the logical file to be connected to the physical file.

Device Support Facilities (ICKDSF). A program used for initialization of DASD volumes and track recovery.

default value. A value assumed when no value has been specified. Synonymous with assumed value.

Device Support Facilities (DSF). An IBM-supplied system control program for performing operations on disk volumes so that they can be accessed by IBM and user programs. Note: Examples of these operations are initializing a disk volume and assigning an alternate track.

device-independent. Pertaining to a program that can be executed successfully without regard for the characteristics of particular types of devices. Contrast with device-dependent.

DFSMS environment. An environment that helps automate and centralize the management of storage. This is achieved through a combination of hardware, software, and policies. In the DFSMS environment for MVS, this function is provided by MVS/ESA SP and DFSMS/MVS, DFSORT, and RACF. See also system-managed storage.

DFSMSdfp. A DFSMS/MVS functional component that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss. A DFSMS/MVS functional component used to copy, move, dump, and restore data sets and volumes.

DFSMShsm. A DFSMS/MVS functional component used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

DFSMShsm-managed volume. (1) A primary storage volume, which is defined to DFSMShsm but which does not belong to a storage group. (2) A volume in a storage group, which is using DFSMShsm automatic dump, migration, or backup services. Contrast with system-managed volume and DFSMSrmm-managed volume.

DFSMShsm-owned volume. A storage volume on which DFSMShsm stores backup versions, dump copies, or migrated data sets.

DFSMS/MVS. An IBM licensed program that together with MVS/ESA SP compose the base MVS/ESA operating environment. DFSMS/MVS consists of DFSMSdfp, DFSMSdss, DFSMShsm, and DFSMSrmm.

direct access. (1) The capability to obtain data from a storage device, or to enter data into a storage device, in a sequence independent from their relative position, by means of addresses indicating the physical position of the data. (2) Contrast with serial access.

direct access storage device (DASD). A device in which access time is effectively independent of the location of the data. Usually disk storage.

directory. (1) A type of file containing the names and controlling information for other files or other directories. (2) An index that is used by a control program to locate one or more blocks of data that are stored in separate areas of a data set in direct access storage.

disk file. A set of related records on disk that are treated as a unit.

distributed data. In SAA usage, data that is split across two or more linked systems but which can be accessed and processed as if it resided on one.

Distributed Data Management (DDM). A feature of the System Support Program Product that allows an application program to work on files that reside in a remote system.

DL/I. See Data Language One.

DLIB. Distribution library. IBM-supplied partitioned data sets on tape containing one or more components that the user restores to disk for subsequent inclusion in a new system.

double-byte character set (DBCS). A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

DSCB. See data set control block.

dsname. data set name. The name of a data set (1 - 40 characters) on the DD statement in the JCL or the dsname operand of the TSO ALLOC command.

dynamic address translation (DAT). In System/390 virtual storage systems, the change of a virtual storage address to a real storage address during execution of an instruction.

dynamic storage. A device that stores data in a manner that permits the data to move or vary with time such that the specified data are not always available for recovery. Magnetic drum and disk storage are dynamic nonvolatile storage. An acoustic delay line is a dynamic volatile storage.

E

emulation. (1) The use of a data processing system to imitate another data processing system, so that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system. Emulation is usually achieved by means of hardware or firmware. (2) The use of programming techniques and special machine

features to permit a computing system to execute programs written for another system.

emulator. A combination of programming techniques and special machine features that permits a computing system to execute programs written for a different system. See also integrated emulator, terminal emulator.

entry-sequenced data set. In OS/390 programs with VSAM, a data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

error recovery procedures (ERP). Procedures designed to help isolate and, where possible, to recover from errors in equipment. The procedures are often used in conjunction with programs that record information on machine malfunctions.

ESCON. Enterprise Systems Connection - A set of products and services that provides a dynamically connected environment using optical cables as a transmission medium.

ESCON Director. A device that provides connectivity capability and control for attaching any two links to each other.

Ethernet. A 10-megabit baseband local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD).

event control block (ECB). A control block used to represent the status of an event.

exit routine. Either of two types of routines: installation exit routines or user exit routines. Synonymous with exit program.

expiration date. The date at which a file is no longer protected against automatic deletion by the system.

extent. Continuous space on a disk or diskette that is occupied by or reserved for a particular data set, data space, or file.

external side. The receiver of a data entity. A module or routine that accepts a 2- or 4-digit-date format entity for further processing from another module or routine.

F

file.

- In PSF/MVS, a member of a partitioned data set or a sequential data set
- In PSF/VSE, a member in a library.sublibrary

file control table (FCT). A table containing the characteristics of files processed by CICS file management.

file name. (1) A name assigned or declared for a file. (2) The name used by a program to identify a file.

fixed window. A technique to determine the century (high-order digits) of a year when represented by two digits. The 2-digit year is compared against a hardcoded threshold. The century designation is limited to a 100-year range spanning only two centuries. For example, assume the threshold is 60, then if the 2-digit year is ≥60, the year is in the 20th century; if the 2-digit year is <60, the year is in the 21st century.

fixed-length record. A record having the same length as all other records with which it is logically or physically associated. Contrast with variable-length record.

foreground partition. In VSE, a space in virtual storage in which programs are executed under control of the system. By default, a foreground partition has a higher processing priority than the background partition.

formatted dump. A dump in which certain data areas are isolated and identified.

forward recovery. (1) The reconstruction of a file by updating an earlier version with data recorded in a journal. (2) The process of reconstructing a file from a particular point by restoring a saved version of the file and then applying changes to that file in the same order in which they were originally made.

G

GDG. See generation data group.

generation data group (GDG). A feature of the MVS catalog that allows a collection of data sets to be kept in chronological order: each data set is called a generation data set.

generation data set. One generation of a generation data group.

Gregorian calendar. Today's general-use calendar of 12 months and 365 days that employs the current leap year algorithm (refer to **Leap year** below).

GRS. global resource serialization. A component of MVS/ESA SP used for sharing system resources and for converting DASD reserve volumes to data set enqueues.

Guest Operating System (GOS). A second operating system that runs on the primary operating system. An example the second operating system is VSE and/or OS/390 and/or TPF etc. running on VM/ESA. In this example VSE, OS/390, TPF, etc. are referred to as a second level system or a VSE guest, OS/390 guest, TPF guest and so on.

Guest Support. A set of functions and services available on the VM/ESA product that allow other operating systems such as OS/390, VSE, TPF, VM, and others to run on the primary host VM/ESA system. This is also sometimes referred to as ″using VM as a hypervisor for running other operating systems″.

H

hardcopy log. In systems with multiple console support or a graphic console, a permanent record of system activity.

HCD. Hardware Configuration Definition. An interactive interface in MVS and OS/390 that enables an installation to define hardware configurations from a single point of control.

HFS data set. hierarchical file system data set. A data set that contains a POSIX-compliant hierarchical file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using the OpenEdition MVS facilities.

high-level language (HLL). A programming language whose concepts and structures are convenient for human reasoning; for example, Pascal. High-level languages are independent of the structures of computers and operating systems.

HMC. Hardware Management Console A console used to monitor and control hardware such as the System/390 microprocessors.

host system. (1) A data processing system used to prepare programs and operating environments for use on another computer or controller. (2) The data processing system to which a network is connected and with which the system can communicate.

I

I/O area. An area of storage that contains data which is used in input/output operations; for example, an I/O buffer.

ICCF. See Interactive Computing Control Facility.

ICKDSF. See Device Support Facilities.

IDCAMS. Utility program name for Access Method Services (AMS).

IMS/VS. See Information Management System/Virtual Storage.

Information Management System/Virtual Storage (IMS/VS). A data base/data communication (DB/DC) system capable of managing complex data bases and networks

information processing. The systematic performance of operations on information in conjunction with a computer system to obtain, manipulate, duplicate, exchange, or communicate its meaning; for example, file management, word processing, document interchange, facsimile, videotext.

Information/Management. A feature of the Information/System licensed program that provides interactive systems management applications for problem, change, and configuration management.

Information/System. In the NetView program, an interactive retrieval program with related utilities designed to provide systems programmers with keyword access to selected technical information contained in either of its companion products, Information/MVS or Information/VM-VSE.

initiator/terminator. The job scheduler function that selects jobs and job steps to be executed, allocates input/output devices for them, places them under task control, and at completion of the job, supplies control information for writing job output on a system output unit.

input data set. A data set that contains data to be processed.

input file. A file that has been opened in order to allow records to be read. Contrast with output file.

input/output (I/O). Pertaining to a device, process, or channel involved in data input, data output, or both.

installation. (1) In system development, preparing and placing a functional unit in position for use. (2) A particular computing system, including the work it does and the people who manage it, operate it, apply it to problems, service it, and use the results it produces.

installation exit. The means specifically described in an IBM software product′s documentation by which an IBM software product may be modified by a customer′s system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace one or more existing modules of an IBM software product, or to add one or more modules or subroutines to an IBM software product, for the

purpose of modifying or extending the functions of the IBM software product.

integer date. A count of days since a specified date. Various IBM software products have defined integer dates as follows:

Language/Product	Days Since
C	1969-Dec-31
COBOL	1600-Dec-31
Language Environment	1582-Oct-14
MVS/CICS/DB2	1899-Dec-31

integrity. The protection of systems, programs, and data from inadvertent or malicious destruction or alteration. See application integrity, data integrity, system integrity.

Interactive Computing and Control Facility (ICCF). An IBM licensed program that makes the services of a VSE-controlled computing system available to authorized display station users.

interactive partition. In VSE, an area of virtual storage dynamically allocated for the purpose of processing a job that was submitted interactively from a terminal.

Interactive Problem Control System (IPCS). A component of VM that permits online problem management, interactive problem diagnosis, online debugging for disk-resident CP abend dumps, problem tracking, and problem reporting.

Interactive System Productivity Facility. An IBM licensed program that serves as a full-screen editor and dialogue manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

internal side. The creator or manipulator of a data entity. Used in this document to mean a module or routine that externalizes a 2- or 4-digit-year format entity to another module or routine.

Internet. A wide area network connecting thousands of disparate networks in industry, education, government, and research. The Internet network uses TCP/IP as the standard for transmitting information.

interoperability. The capability to communicate, execute programs, or transfer data among various functional units in a way that requires the user to have little or no knowledge of the unique characteristics of those units.

IOCDS. An input/output configuration data set (IOCDS) contains different configuration definitions for the selected processor. Only one IOCDS is used at a time. The IOCDS contains I/O configuration data on the files associated with the processor controller on the host processor, as it is used by the channel subsystem. The CSS uses the configuration data to control I/O requests. The IOCDS is built from the production IODF.

IOCP. An IOCP (I/O configuration program) is the hardware utility that defines the hardware I/O configuration to the channel subsystem. For this definition IOCP retrieves information about the following: the channel paths in the processor complex, control units attached to the channel paths, and I/O devices assigned to the control unit.

IODF. An IODF (input/output definition file) is a VSAM linear data set that contains I/O definition information. This information includes processor I/O definitions (formerly specified by IOCP input streams) and operating system I/O definitions (formerly specified by MVSCP input streams). A single IODF can contain several processor and several operating system I/O definitions.

IPDS. Intelligent Printer Data Stream. An architected host-to-printer data stream that contains both data and controls defining how the data is to be presented.

IPL. Initial Program Load. (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage, as at the beginning of a work day or after a system malfunction or as a means to access updated parts of the system. (3) The process of loading system programs and preparing a system to run jobs.

J

JCL. Job Control Language. A sequence of commands used to identify a job to an operating system and to describe a job's requirements.

JECL. Job Entry Control Language - also referred to as JES2 or JES3 control statements that are submitted with a job's JCL.

JES. Job Entry Subsystem. A system facility for spooling, job queueing, and managing the scheduler work area.

job accounting. A function that collects information pertaining to how a job uses system resources.

job control. In VSE, a program called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to symbolic names, set switches for program use, log (or print) job control statements, and fetch the first phase of each job step.

job control language (JCL). A control language used to identify a job to an operating system and to describe the job's requirements.

Job Entry Subsystem. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

job step. You enter a program into the operating system as a job step. A job step consists of the job control statements that request and control execution of a program and request the resources needed to run the program. A job step is identified by an EXEC statement. The job step can also contain data needed by the program. The operating system distinguishes job control statements from data by the contents of the record.

job stream. The sequence of representation of jobs or parts of jobs to be performed, as submitted to an operating system. Synonymous with input stream, run stream.

journaling. The process of recording changes made in a physical file member in a journal. Journaling allows the programmer to reconstruct a physical member by applying the changes in the journal to a saved version of the physical file member.

Julian date. As a general term used widely in computer programming and this document: A date in the format YYDDD. A date format that contains the year in positions 1 and 2, and the day in positions 3 through 5. The day is represented as 1 through 366, right adjusted, padded with zeroes on the left. For example, 1996-August-29 is 96242.

However, the above definition is accurately referred to as the **Ordinal Day of Year** date, and an accurate definition of **Julian Day Number** is as follows:

The astronomical system that counts the days since the First of January in the year 4713 BCE (the year −4712 before the common era). This scheme was invented by the astronomer Joseph Scaliger in the 16th century and named by him for his father Julius. The leap year reforms implicit in the new scheme were adopted at the command of Pope Gregory XIII in the year 1582 - hence the Gregorian Calendar. The Gregorian, or New Style, calendar was adopted in Britain and their colonies in September of 1752. (September of that year was missing 11 days. The 14th followed the 2nd.)

The remainder left when dividing the Julian Day Number by 7 indicates the day of week of the specified date. Zero corresponds to Monday, 1 to Tuesday, up through 6 for Sunday.

For example, 1996-Aug-29 is equivalent to Julian Day 2450325. Further, the hour of the day is expressed as a decimal such that 2450325.5 is midnight 1996-Aug-29, based on the fact that a Julian Day begins at midday (noon).

K

key field. (1) In VSAM, a field, located in the same position in each record of a file or data set, whose content is used for the key of a record. (2) In IMS/VS, the field in a database segment used to store segment occurrences in sequential ascending order. A key field is also a search field. Synonymous with sequence field.

key-sequenced data set (KSDS). A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

L

label area. Synonym for label information area

label information area. In VSE, an area on a direct access storage device that stores label information read from job control statements or commands. Synonymous with label area.

Leap year. A year either evenly divisible by 400 **or** evenly divisible by 4 and not evenly divisible by 100. For example, the years 1700, 1800, 1900, and 1995 are not leap years, but the years 1600, 1996, and 2000 are leap years.

librarian. In VSE, the set of programs that maintains, services, and organizes the system and private libraries.

library member. In VSE, the smallest unit of data that can be stored into and retrieved from a sublibrary.

licensed program (LP). A separately priced program and its associated materials that bear an IBM copyright and are offered to customers under the terms and conditions of either the Agreement for IBM Licensed Programs (ALP) or the IBM Program License Agreement (PLA).

Lilian date. The number of days since 1582-October-14. 1582-October-15 is Lilian day 1, 1582-October-16 is Lilian day 2, and so on. (Named for Aloysius Lilius (an advisor to Pope Gregory XIII) who, together with his brother, constructed the current Gregorian calendar.)

link pack area (LPA). In OS/390, an area of main storage containing reenterable routines from system libraries. Their presence in main storage saves loading time.

linkage editor. A program that resolves cross-references between separately assembled object modules and then assigns final addresses to create a single relocatable load module. The linkage editor then stores the load module in a program library in main storage.

link-edit. To create a loadable computer program by means of a linkage editor.

load module. An application or routine in a form suitable for execution. The application or routine has been compiled and link-edited; that is, address constants have been resolved.

load module library. A partitioned data set used to store and retrieve load modules. See also object module library, source module library.

local area network (LAN). (1) A computer network located on a user′s premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. See also wide area network. See figure 88. ñNote: A LAN does not use store and forward techniques. (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network. See also token ring.

lock. (1) A serialization mechanism by means of which a resource is restricted for use by the holder of the lock. See exclusive lock, shared lock. (2) The means by which integrity of data is ensured by preventing more than one user from accessing or changing the same data or object at the same time.

log data set. A data set consisting of the messages or message segments recorded on auxiliary storage by the ACF/TCAM logging facility.

logical device. (1) A file for conducting input or output with a physical device. (2) A file for mapping user I/O between virtual and real devices.

logical record. (1) A set of related data or words considered to be a record from a logical viewpoint. (2) A record from the standpoint of its content, function, and use rather than its physical attributes, that is, a record defined in terms of the information it contains. (3) In CICS/VS, a data record sent by one transaction program to another. The length of the record is contained in a two-byte field immediately preceding the record. (4) In VSAM, a unit of information normally pertaining to a single subject; a logical record is the user record requested of or given to the data management function. (5) In COBOL, the most inclusive data item.

M

main task. In VSE, the main program within a partition in a multiprogramming environment.

maintenance. Any activity intended to retain a functional unit in, or to restore it to, a state in which it can perform its required function. Maintenance includes keeping a functional unit in a specified state by performing activities such as tests, measurements, replacements, adjustments, and repairs.

MAS. Multi-Access Spool facility. A loosely connected complex of JES2 members.

mass conversion. An automated process that includes program translation, JCL conversion, and file transfer. This process is what makes possible the rapid switchover from VSE to MVS.

master console. In a system with multiple consoles, the basic console used for communication between the operator and the system.

MCS. (1) Multiple Console Support. A feature of MVS that permits selective message routing to up to 32 operator′s consoles. (2) modification control statement. An SMP/E control statement used to package a SYSMOD. These statements describe the elements of a program and the relationships that program has with other programs that may be installed on the same system.

menu. A list of options displayed to the user by a data processing system, from which the user can select an action to be initiated.

message data set. A data set on disk storage that contains queues of messages awaiting transmission to particular terminal operators or to the host system.

migrate. To move to a changed operating environment, usually to a new release or version of a system.

migration (VSE/MVS). The entire process of transition from a VSE environment to an MVS environment. Migration includes training, project planning and management, system and configuration setup, conversion design, and the conversion itself.

minidisk. Synonym for virtual disk

module. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine.

multiprocessor. (1) A computer including two or more processors that have common access to a main storage. (2) A system of two or more processing units, ALUs, or processors that can communicate without manual intervention.

multitasking. A mode of operation that provides for concurrent performance, or interleaved execution of two or more tasks.

multithreading. Pertaining to concurrent operation of more than one path of execution within a computer.

multivolume file. A file contained on more than one storage medium.

N

NetView DM. IBM NetView Distribution Manager.

network definition. In VTAM, the process of defining the identities and characteristics of each node in the network and the arrangement of the nodes in that system.

network management. The process of planning, organizing, and controlling a communications-oriented system.

network resource. In ACF/VTAM, a network component such as a local network control program, SDLC data link, or peripheral node. In multiple-domain networking, cross-domain resource managers (CDRMs) and logical units (LUs) in other domains are also network resources.

network topology. The schematic arrangement of the links and nodes of a network.

node. An endpoint of a link or a junction common to two or more links in a network. Nodes can be processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities. (6) In VTAM, a point in a network defined by a symbolic name. See major node, minor node.

nonstandard labels. Labels that do not conform to American National Standard or IBM standard label conventions.

O

Object Access Method (OAM). In the IBM ImagePlus system, a program that provides object storage, object retrieval, and object storage hierarchy management. The Object Access Method isolates applications from storage devices, storage management, and storage device hierarchy management.

object code. Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code.

object module. (1) All or part of an object program sufficiently complete for linking. Assemblers and compilers usually produce object modules. (2) A set of instructions in machine language produced by a compiler from a source program.

object program. (1) A target program suitable for execution. An object program may or may not require linking. (2) Contrast with source program.

open. The function that connects a file to a program for processing.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible.

operator console. A display console used for communication between the operator and the system, used primarily to specify information concerning application programs and I/O operations and to monitor system operation.

option. A specification in a statement that may be used to influence the execution of the statement.

Ordinal Day of Year. See **Julian Date**

output area. An area of storage reserved for output.

output class. In OS/390, one of up to 36 different categories, defined at an installation, to which output data produced during a job step can be assigned. When an output writer is started, it can be directed to process from one to eight different output data classes.

output data set. A data set that contains data that is to be printed or displayed.

output file. (1) A file that has been opened in order to allow records to be written. (2) Contrast with input file.

overlay structure. A graphic representation showing the relationships of segments of an overlay program and how the segments are arranged to use the same main storage area at different times.

P

parameter list. A list of values that provides a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

partition. In VSE, a division of the virtual address area that is available for program execution.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. Synonymous with program library.

Pascal. A high-level, general-purpose programming language, related to ALGOL. Programs written in Pascal are block structured, consisting of independent

routines. They can run on different computers with little or no modification.

PDS. partitioned data set. A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. Contrast with sequential data set.

PDSE. partitioned data set extended. A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

peer. (1) In network architecture, any functional unit that is in the same layer as another entity. (2) A corresponding node or entity.

physical IOCS (PIOCS). Supervisory routines that schedule and supervise the execution of channel programs. Physical IOCS controls the actual transfer of records between external storage and main storage, and provides I/O device error recovery.

physical record. (1) A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may contain all or part of one or more logical records. (2) The amount of data transferred to or from auxiliary storage. Synonymous with block.

portability. (1) The capability of a program to be executed on various types of data processing systems without converting it to a different language and with little or no modification. (2) The ability to run a program on more than one computer without modifying it. (3) Synonymous with transportability.

POWER. A VSE program used to spool input and output. It also allows exchange of files with, or job runs on, a remote processor Originally an acronym for Priority Output Writer Execution and Reader.

procedure. A named block of code that can be invoked, usually via a call.

PR/SM. Processor Resource / Systems Manager

precompile. To process programs containing SQL statements before they are compiled. SQL statements are replaced with statements that will be recognized by the host language compiler. The output from this precompile includes source code that can be submitted to the compiler and used in the bind process.

preventive maintenance. (1) Maintenance performed specifically to prevent faults from occurring. (2) Contrast with corrective maintenance.

private address space. An address space assigned to a particular user.

problem determination. The process of determining the source of a problem; for example, a program component, machine failure, telecommunication facilities, user or contractor-installed programs or equipment, environmental failure such as a power loss, or user error.

procedural language. A programming language in which computations are expressed in terms of statement sequences; for example, Pascal. Synonym for procedure-oriented language.

procedure. In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call.

procedure library. A program library in direct access storage with job definitions. The reader/interpreter can be directed to read and interpret a particular job definition by an execute statement in the input stream.

processor storage. (1) The storage provided by one or more processing units. (2) In virtual storage systems, synonymous with real storage.

PROCLIB. procedure library. A program library in direct access storage with job definitions. The reader/interpreter can be directed to read and interpret a particular job definition by an execute statement in the input stream.

program. A sequence of instructions suitable for processing by a computer. Processing may include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as to execute it.

program offering. An unwarranted licensed program. See also vendor-logo product.

program product. Deprecated term for licensed program.

programmer logical unit. In VSE, a logical unit available primarily for user-written programs.

programming language. An artificial language for expressing computer programs.

programming system. (1) In a programming environment, the software required for the development and use of computer programs. (2) In a data processing system, the software needed to use one or more programming languages.

PSF. Print Services Facility. A licensed program that manages and controls the input data stream and output data stream required by supported IBM page printers. PSF combines print data with other resources and printing controls to produce AFP output. There are PSF products for MVS (OS/390), VSE, RS/6000 and OS2 platforms.

Q

qualified name. (1) A data name explicitly accompanied by a specification of the class to which it belongs in a specified classification system. (2) A name that has been made unique by the addition of one or more qualifiers.

R

RACF. Resource Access Control Facility. An IBM-licensed program that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

read access. In computer security, permission to read information.

real storage. The main storage in a virtual storage system. Physically, real storage and main storage are identical. Conceptually however, real storage represents only part of the range of addresses available to the user of a virtual storage system. Traditionally, the total range of addresses available to the user was provided by the main storage.

real storage management (RSM). Routines that control allocation of pages in real storage.

reason code. A code that identifies the reason for a detected error.

record format. The definition of how data are structured in the records contained in a file. The definition includes record name, field names, and field descriptions, such as length and data type. The record formats used in a file are contained in the file description.

record type. The classification of records in a file.

recorder file. Synonym for system recorder file.

recovery procedure. (1) A process in which a specified data station attempts to resolve conflicting or erroneous conditions arising during the transfer of data. (2) An action performed by the operator when an error occurs to permit processing to continue.

reserved word. (1) In programming languages, a keyword that may not be used as an identifier. (2) A word used in a source program to describe an action to be taken by the program or the compiler. It must not appear in the program as a user-defined name or a system name.

Resource Access Control Facility (RACF). An IBM-licensed program that provides for access control by identifying and by verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

resource definition online (RDO). A CICS interactive facility to create and modify system resources.

response time. The elapsed time between the end of an inquiry or demand on a computer system and the beginning of the response; for example, the length of time between an indication of the end of an inquiry and the display of the first character of the response at a user terminal.

retention period. (1) The length of time for which data on a data medium is to be preserved.

return code. A value returned to a program to indicate the results of an operation requested by that program.

RMF. Resource Measurement Facility. An IBM licensed program that measures selected areas of system activity and presents the data collected in the format of printed reports, system management facilities (SMF) records, or display reports. Use RMF to evaluate system performance and identify reasons for performance problems.

RJE. Remote Job Entry. Submission of job control statements and data from a remote terminal, causing the jobs described to be scheduled and executed as though encountered in the input stream.

rolling window. Synonymous with **sliding window**.

S

SAA environments. Those environments in which IBM intends to provide full implementation of applicable SAA architectural elements.

SAM-VSAM. SAM files in VSAM-managed space.

save area. Area of main storage in which contents of registers are saved.

scheduler. A computer program designed to perform functions such as scheduling, initiation, and termination of jobs.

SDSF. System Display and Search Facility. An IBM licensed program that or element of OS/390 that allows TSO/E users to browse JES2 spooled data sets, and view and manipulate JES2 job queues, and devices.

secondary space allocation. In systems with VSAM, area of direct access storage space allocated after primary space originally allocated is exhausted.

sequential data set. A data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Contrast with direct data set.

sequential file. A file in which records are processed in the order in which they are entered and stored in the file. Contrast with direct file, indexed file.

sequential processing. (1) The processing of logical records in the order in which they are accessed. (2) The processing of records in the order in which they exist in a file. Synonymous with consecutive processing.

ServerPac. A software delivery package consisting of installed products and integrated service for a ready-to-IPL system. To install, you use the CustomPac Installation Dialog -- the same dialog that is used for all the CustomPac offerings, including SystemPac and ProductPac.

shared spooling. A function that permits the VSE/POWER account file, data file, and queue file to be shared among several computer systems with VSE/POWER.

SIE. SoftwareXcel Installation Express. A service offering in the USA built on SystemPac with IBM assistance included for installation and implementation.

simulation. (1) The use of a data processing system to represent selected behavioral characteristics of a physical or abstract system; for example, the representation of air streams around airfoils at various velocities, temperatures, and air pressures. (2) Contrast with emulation.

sliding window. A technique to determine the century (high-order digits) of a year when represented by two digits. The user specifies the number of years (both past and future) within a 100-year window spanning two centuries. For example, assume the window is set at 19 future years (1996-2014) and 80 past years (1915-1994). Dates in the range 00–14 (inclusive) are designated 21st century dates because they fall into the future window. Dates in the range 15–99 (inclusive) fall into the 20th century.

SMF. system management facilities. A component of MVS and OS/390 that collects input/output (I/O) statistics, provided at the data set and storage class levels, which helps you monitor the performance of the direct access storage subsystem.

SMP/E. System Modification Program Extended. SMP/E is the IBM product designed to install new function and subsequent service into target libraries and distribution libraries.

SMS. Storage Management Subsystem. A DFSMS/MVS or MVS/DFP facility used to automate and centralize the management of storage.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network accessible units (NAUs), boundary function, gateway function, and intermediate session routing function components; and the transport network.

software. All or part of the programs, procedures, rules, and associated documentation of a data processing system. Software is an intellectual creation that is independent of the medium on which it is recorded.

source code. The input to a compiler or assembler, written in a source language. Contrast with object code.

source language. The programming language for expressing source programs that a particular translator can accept.

source program. (1) A set of instructions written in a programming language that must be translated to machine language before the program can be run. (2) Contrast with object program.

source statement. A statement written in symbols of a programming language; for example, RPG, COBOL, BASIC, and PL/I specifications are source statements.

spool file. (1) A file that contains output data that has been saved for later processing. (2) One of three VSE/POWER files on disk: queue file, data file, and account file.

spreadsheet. A worksheet arranged in rows and columns, in which a change in the contents of one cell can cause electronic recomputation of one or more cells, based on user defined relations among the cells.

stand-alone. Pertaining to operation that is independent of any other device, program, or system.

standard label. A fixed-format record that identifies a volume of data such as a tape reel or a file that is part of a volume of data.

statement. In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations.

storage location. A position in a storage device that is uniquely specified by means of an address.

Storage Management Subsystem (SMS). A component of MVS/DFP that is used to automate and centralize the management of storage by providing the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

structured programming. A method for constructing programs using only hierarchically nested constructs each having a single entry and a single exit point. Three types of control flow are used in structured programming: sequential, conditional, and iterative.

sublibrary. In VSE, a subdivision of a library. See also library.

subprogram. A program invoked by another program. Contrast with main program.

subroutine. A sequenced set of instructions or statements that may be used in one or more computer programs and at one or more points in a computer program. (3) A group of instructions that can be part of another routine or can be called by another program or routine.

subsystem. A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.

supervisor. The part of a control program that coordinates the use of resources and maintains the flow of processing unit operations. See also system supervisor.

symbolic name. In a programming language, a unique name used to represent an entity such as a field, file, data structure, or label.

syntax. (1) The relationship among characters or groups of characters, independent of their meanings or the manner of their interpretation and use. (2) The rules governing the structure of a language.

syntax error. A compile-time error caused by incorrect syntax. See also semantic error.

sysplex. A multiple-MVS system environment that allows MCS consoles or extended MCS consoles to receive messages and send commands across systems.

system console. A console, usually equipped with a keyboard and display screen, that is used by an operator to control and communicate with a system.

system date. The date established for the system when it is started.

system generation (SYSGEN). The process of selecting optional parts of an operating system and of creating a particular operating system tailored to the requirements of a data processing installation.

system maintenance. The modification of a system to correct faults, to improve performance, or to adapt the system to a changed environment or changed requirements.

system management facilities (SMF). See SMF.

system-managed storage. Storage managed by the Storage Management Subsystem. SMS attempts to deliver required services for availability, performance, space, and security to applications. See also DFSMS environment.

system-managed volume. A DASD, optical, or tape volume that belongs to a storage group. Contrast with DFSMShsm-managed volume and DFSMSrmm-managed volume.

System Modification Program (SMP). A program used to install software and software changes on MVS systems.

system operator. An operator responsible for performing system- oriented procedures.

system programmer. A programmer who plans, generates, maintains, extends, and controls the use of an operating system with the aim of improving overall productivity of an installation.

system resources. Those resources controlled by the system, such as programs, devices, and storage areas that are assigned for use in jobs.

system software. Software that is part of or made available with a computer system and that determines how application programs are run; for example, an operating system. Contrast with application software.

system support. The continued provision of services and material necessary for the use and improvement of an implemented system.

SystemPac. A software package consisting of installed products for a ready-to-IPL system. Some of the products have been customized in response to information provided to IBM. A SystemPac can be used to install an MVS system for the first time or to replace an existing MVS system.

systems management. Functions in the application layer related to the management of Open Systems Interconnection resources and their status across all layers of the OSI architecture.

T

task. (1) In a multiprogramming or multiprocessing environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer. (2) In VSE, the basic unit of synchronous program execution. A task competes with other tasks for computer system resources such as processing time and I/O channels.

Telnet. In TCP/IP, an application protocol that allows a user at one site to access a remote system as if the user′s display station were locally attached. Telnet uses the Transmission Control Protocol as the underlying protocol.

temporary data set. A data set that is created and deleted in the same job. Contrast with nontemporary data set.

temporary storage. In computer programming, storage locations reserved for intermediate results. Synonymous with working storage.

test plan. A plan that establishes detailed requirements, criteria, general methodology, responsibilities, and general planning for test and evaluation of a system.

Time Sharing Option (TSO). An operating system option; for the System/370 system, the option provides interactive time sharing from remote terminals.

token-ring network. A network that uses a ring topology, in which tokens are passed in a circuit from node to node. A node that is ready to send can capture the token and insert data for transmission.

transaction processing. A sequence of operations on a database that is viewed by the user as a single, individual operation.

transient. Pertaining to a program or subroutine that does not reside in main storage or to a temporary storage area for such a program.

transparent. (1) Pertaining to operations or data that are of no significance to the user. (2) In data transmission, pertaining to information not recognized by the receiving program or device as transmission control characters.

transportability. Synonym for portability.

TSCF. Target System Control Facility. Part of System Automation OS/390 which uses NetView to allow a host OS/390 system to automate operations at target systems.

TSO/E. time sharing option/extended. An option on the operating system; for System/370, the option provides interactive time sharing from remote terminals.

U

unit address. The three-character address of a device, specified at the time a system is installed; for example, 191 or 293.

unit of work. In advanced program-to-program communications, the amount of processing that is started directly or indirectly by a program on the source system.

update authority. The ability to add, change, or cancel items.

upward compatibility. The capability of a computer to execute programs written for another computer without major alteration, but not vice versa.

user catalog. See VSAM user catalog.

user exit. A programming service provided by an IBM software product that may be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

user identification (user ID). (1) A string of characters that uniquely identifies a user to a system. (2) The name used to associate the user profile with a user when a user signs on the system.

user interface. (1) Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device. (2) In SAA usage, any of the actions or items defined by Common User Access (CUA) architecture that allow a user to interact with and perform operations on a computer.

user profile. In computer security, a description of a user that includes such information as user ID, user name, password, access authority, and other attributes obtained at logon.

user program. A user-written program.

user-defined word. In COBOL, a word that must be supplied by the user to satisfy the format of a clause or statement.

utility program. A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program. Synonymous with service program.

V

verification test. A test of a system to prove that it meets all its specified requirements at a particular stage of its development.

virtual address. The address of a location in virtual storage. A virtual address must be translated into a real address in order to process the data in processor storage.

virtual address space. (1) In virtual storage systems, the virtual storage assigned to a batched or terminal job, a system task, or a task initiated by a command. (2) In VSE, a subdivision of the virtual address area available to the user for the allocation of private, nonshared partitions.

virtual device. A device that appears to the user as a separate entity, but is actually a shared portion of a real device; for example, several virtual terminals can exist simultaneously, but only one is active at any given time.

virtual disk. (1) Main storage used as if it were a disk device. (2) In VM, a physical disk storage device, or a logical subdivision of a physical disk storage device, that has its own address, consecutive storage space for data, and index or description of stored data so that the data can be accessed.

virtual machine (VM). A virtual data processing system that appears to be at the exclusive disposal of a particular user, but whose functions are accomplished by sharing the resources of a real data processing system.

virtual storage. The storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations.

Virtual Storage Access Method (VSAM). An access method for indexed or sequential processing of fixed and variable-length records on direct access devices.

volume label. An area on a standard label tape used to identify the tape volume and its owner. This area is the first 80 bytes and contains VOL 1 in the first four positions.

volume serial number. A number in a volume label assigned when a volume is prepared for use in a system.

VSAM managed space. A user-defined space on disk that is under the control of VSAM.

VSE (Virtual Storage Extended). Any of the VSE operating systems and environments.

W

warm start. (1) A restart that allows reuse of previously initialized input and output work queues. (2) In VM, the result of an initial program load (IPL) that does not erase previous system data.

work file. (1) A file used for temporary storage of data being processed. (2) In sorting, an intermediate file used for temporary storage of data between phases.

work space. That portion of main storage that is used by a computer program for temporary storage of data. Synonymous with working space.

write access. In computer security, permission to write to an object.

Y

Year2000 challenge. The potential problems and its variations that might be encountered in any level of computer hardware or software from microcode to application programs, files, and databases that need to correctly interpret year-date data represented in 2-digit-year format caused by the transition to the year 2000.

Year2000 ready. The capability of a Product, when used in accordance with its associated documentation, to correctly process, provide and/or receive date data within and between the 20th and 21st centuries, provided that all products (for example, hardware, software, and firmware) used with the Product properly exchange accurate date data with it.

Year2000 support. The ability to provide **Year2000 readiness**.

Year2000 transition. The process of revising systems and databases) to correctly process date data both within and between the 20th and 21st centuries.

YY format. Synonymous with **2-digit-year format**.

YYYY format. Synonymous with **4-digit-year format** and a subset of **CCYY format**.

List of Abbreviations

ABEND	ABnormal END
ACB	Access Control Block
ACF/NCP	Advanced Communications Facility/Network Control Program
ACF/SSP	Advanced Communications Facility/???
ACIF	AFP Conversion and Indexing Facility
ACS	Automatic Class Selection
ADSTAR	ADvanced STorage And Retrieval
AFF	AFFinity
AFP	Advanced Function Presentation
AFPDS	Advanced Function Printing Data Stream
AIX	Advanced Interactive eXecutive
AMA	Automatic Message Accounting
AMODE	Addressing MODE
AMS	Access Method Services
ANSI	American National Standards Institute
AOR	Application Owning Region
APAR	Authorized Program Analysis Report
APF	Authorized Program Facility
API	Application Program Interface
APPC	Advanced Program-to-Program Communication
APPL	APPLication
APPLID	APPLication IDentifier
APPN	Advanced Peer-to-Peer Networking
AR	Application Requester
ASCII	American National Standard Code for Information Interchange
ASID	Address Space IDentifier
ATM	Asynchronous Transfer Mode
BCD	Binary Coded Decimal
BCP	Basic Control Program
BCS	Basic Catalog Structure
BDAM	Basic Direct Access Method
BDT	Batch Data Interface
BG	BackGround
BISAM	Basic index Sequential Access Method
BLL	Base Locator for Linkage
BLP	Bypass Label Processing
BOS	Basic Operating System
BPAM	Basic Partitioned Access Method
BSAM	Basic Sequential Access Method
BSC	Binary Synchronous Communication
BSD	Berkeley Software Distribution
BSF	Back Space File
BSL	Basic Systems Language
BSR	Back Space Record
BTAM	Basic Telecommunications Access Method
BTAM-ES	BTAM-Extended Support
BWO	Backup-While-Open
C++	A programming language, a preprocessor to C
CA	Control Area
CAT	CATalog
CBIPO	Custom Built Initial Program Offering
CBPDO	Custom-Built Product Delivery Option
CCB	Channel Command Block
CCH	Channel Check Handler
CCCA	Cobol CICS Conversion Aid
CCW	Channel Command Word
CD	Compact Disc
CD-ROM	Compact Disk - Read Only Memory
CEC	Central Electronics Complex
CEE	Common Execution Environment
CEMT	CICS Master Terminal Transaction

CGI	Common Gateway Interface
CHKPT	CHecKPoinT
CI	Control Interval
CICS	Customer Information Control System
CICS/DOS/VS	Customer Information Control System/Disk Operating System/Virtual Storage
CICS/VS	Customer Information Control System/Virtual Storage
CLIST	Command LIST
CMIP	Common Management Information Protocol
CMOS	Complementary Metal Oxide Semiconductor
CMS	Conversational Monitor System
COBOL	COmmon Business Oriented Language
CP	Control Program
CPU	Central Processing Unit
CRLF	Carriage Return/Line Feed
CSA	Common System Area
CSAR	Complex System Availability and Recovery
CSD	CICS System Definition
CSECT	Control SECTion
CSF	Crypto Support Facility
CSL	Callable Services Library
CSM	Communication Storage Manager
CSP	Cross System Product
CSSF	Customer Software Support Facility
CTC	Channel To Channel
CTCA	Channel To Channel Adapter
CVOL	Control Volume
DA	Direct Access
DADSM	Direct Access Device Space Management
DAM	Direct Access Method
DASD	Direct Access Storage Device
DB	Data Base
DB/DC	Data Base/Data Communications
DB2	Data Base 2
DBA	Data Base Administrator
DBCS	Double Byte Character Set
DBD	Data Base Directory
DBRC	Data Base Recovery Control
DBSU	Data Base Services Utility
DC	Data Communication
DCB	Data Control Block
DCE	Distributed Computing Environment
DCF	Document Composition Facility
DCT	Destination Control Table
DD	Data Definition, Data Dictionary
DDL	Data Definition Language
DDNAME	Data Definition NAME
DEB	Data Extent Block
DECB	Data Event Control Block
DEQ	DEQueue
DES	Data Encryption Standard
DFDSS	Data Facility Data Set Services
DFHSM	Data Facility Hierarchical Storage Manager
DFP	Data Facility Product
DFS	Distributed File System
DFSMS	Data Facility Storage Management Subsystem
DISOSS	DIStributed Office Support System
DITS	Data Information Transfer Set
DITTO	Data Interfile Transfer, Testing & Operations utility
DL/I	Data Language 1
DLBL	Disk LaBeL
DLIB	Distribution LIBrary
DML	Data Manipulation Language
DOS	Disk Operating System
DOS/VS	Disk Operating System/Virtual Storage
DP	Data Processing
DQM	Distributed Queue Management
DRDA	Distributed Relational Database Architecture
DS	Define Storage
DSA	Dynamic Storage Area

DSCB	Data Set Control Block
DSECT	Dummy Control SECTion
DSN	Data Set Name
DSNAME	Data Set NAME
DSNX	Distributed Systems Node eXecutive
DSS	Data Set Services
DTF	Define The File
DTL	Define The Lock
EBCDIC	Extended Binary Coded Decimal Interchange Code
ECB	Event Control Block
ECSA	Extended Common Service Area
EMIF	ESCON Multiple Image Facility
ENDREQ	END REQuest
ENQ	ENQueue
EOD	End Of Data
EOF	End Of File
EOJ	End Of Job
EOP	End Of Page
EP	Emulation Program
EQU	EQUate
EREP	Environmental error Record Editing and Printing program
ERP	Enterprise Resource Planning
ESA	Enterprise Systems Architecture
ESCON	Enterprise Systems CONnection
ESD	External Symbol Dictionary
ESDS	Entry Sequenced Data Set
ESTAE	Extended Specify Task Abnormal Exit
ETXR	End-of-Task eXit Routine
EXCP	EXecute Channel Program
EXLST	EXit LiST
FB	Fixed Block
FBA	Fixed Block Architecture
FCB	Forms Control Buffer
FCT	File Control Table
FD	File Definition
FDDI	Fiber Distributed Data Interface
FEOV	Force End of Volume
FILESEC	FILE SECurity
FIPS	Federal Information Processing Standard
FIT	Fast Implementation Techniques
FM	File Mode
FORMDEF	FORMat DEFinition
FORTRAN	FORmula TRANslation
FSA	Functional Subsystem Application
FSF	Forward Space File
FSI	Full Screen Interface
FSR	Forward Space Record
FSS	Functional SubSystem
FTP	File Transfer Program
GB	GigaByte
GCS	Group Control System
GDDM	Graphical Data Display Manager
GDG	Generation Data Group
GFS	Get File Storage
GHN	Get Hold Next
GHU	Get Hold Unique
GML	Generalized Markup Language
GN	Get Next
GRS	Global Resource Serialization
GSAM	Global Shared Access Method
GSR	Global Shared Resources
GTF	Generalized Trace Facility
GU	Get Unique
GUI	Graphical User Interface
HCD	Hardware Configuration Definition
HD	Hierarchic Direct
HDAM	Hierarchic Direct Access Method
HEX	HEXadecimal
HFS	Hierarchical File System
HIDAM	Hierarchic Indexed Direct Access Method
HISAM	Hierarchic Indexed Sequential Access Method
HLL	High Level Language
HMC	Hardware Management Console

HPR High Performance Routing

HSM Hierarchical Storage Manager

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

I/O Input/Output

I/S Information Systems

I/T Information Technology

IBM International Business Machines

ICA Integrated Communications Adapter

ICCF Interactive Computing and Control Facility

ICF Interactive Command Facility

ICI Improved Control Interval

ICIP Improved Control Interval Processing

ICKDSF Device Support Facilities

ICSF Integrated Cryptographic Service Facility

ICSS Internet Connection Secure Server

ID IDentification/IDentifier

IDCAMS The Program Name for Access Method Services

IEBCOPY Utility Program

IEBGENER Utility Program

IGS Interactive Graphics System

II Interactive Interface

ILC Instruction Length Code

IMF Information Management Facility

IML Initial Microprogram Load

IMS Information Management System

IMS/DB Information Management System/Data Base

IMS/VS Information Management System/Virtual Storage

IOB Input/Output Block

IOCP I/O Channel Program

IOCS Input/Output Control System

IOS Input/Output System

IP Internet Protocol

IPCS Interactive Problem Control System

IPDS Intelligent Printer Data Stream

IPF Interactive Productivity Facility

IPL Initial Program Load

ISA Initial Storage Area

ISAM Indexed Sequential Access Method

ISC Integrated Storage Control

ISMA Information Systems Management Architecture

ISMF Interactive Storage Management Facility

ISPF Interactive System Productivity Facility

ISV Independent Software Vendor

ITSO International Technical Support Organization

IUG Interactive Utility Generation

IVP Implementation Verification Program

JCL Job Control Language

JECL Job Entry Control Language

JES Job Entry Subsystem

JES2 A functional Extension of the HASP II Program

JES3 A functional Extension of the ASP Program

KANJI A character set of symbols used in Japanese Ideographic Alpha

KSDS Key Sequenced Data Set

LAN Local Area Network

LANRES Local Area Network Resource Extension and Services

LCHILD Logical CHILD

LCP Language Conversion Program

LE Linkage Editor

LIBR LIBRarian

LIC Licensed Internal Code

LIFO Last In First Out

LIOCS Logical Input/output Control System

LNKLST Link Library Concatenation

LOGON Log On

LP Logical Partition

LPA Link Pack Area

LPAR	Logically PARtitioned mode
LRECL	Logical RECord Length
LRU	Least Recently Used, Line Replaceable Unit
LSPR	Large Systems Performance Reference
LSR	Local Shared Resources
LTM	Local Transport Mechanism
LU	Logical Unit
LUP	Logical User Profile
MAPS	SNA Network Interconnecting Package
MAS	Multi-Access Spool
MB	MegaByte
MCS	Multiple Console Support
MCSOPER	MVS Console Support OPERator
MCT	Master Control Table
MGCR	Master Get Command Routine
MIH	Missing Interruption Handler
MPF	Message Processing Facility
MQI	Message Queue Interface
MQSERIES	Messaging and Queuing SERIES
MRO	Multi-Region Operation
MS	Migration System
MSHP	Maintain System History Program
MVS	Multiple Virtual Storage
MVS/BDT	Multiple Virtual Storage/Bulk Data Transfer
MVS/DFP	Multiple Virtual Storage/Bulk Data Transfer
MVS/ESA	Multiple Virtual Storage/Enterprise Systems Architecture
MVS/RSU	MVS Recommended Service Upgrade
NCP	Network Control Program
NDF	Network Definition Facility
NFS	Network File System
NJE	Network Job Entry
NSR	Network Service Request
OC	Operator Communication
OCCF	Operator Communications Control Facility
OCO	Object Code Only
OE	Order Entry
OEM	Original Equipment Manufacturer
OGL	Overlay Generation Language
OLPD	On-Line Problem Determination
ONC	Open Network Computing
OPC	Operations, Planning & Control
OPR	OPeRations
OPTCD	OPtional Control Program Service
OS	Operating System
OS/2	Operating System/2
OS/390	Operating System/390
OS/VS	Operating System/Virtual Storage
OSA	Multi-Access Spool
OSA	Open Systems Adapter
OSF	Operational Support Facility
OUTLIM	OUTput LIMiting facility
PARMLIB	PARaMeter LIBrary
PC	Personal Computer
PCB	Program Control Block
PCCU	Primary Communication Control Unit
PCLK	Personal Computer Link Feature
PCT	Program Control Table
PDB	Page Description Block
PDF	Process Control Block
PDS	Partitioned Data Set
PEND	POWER END
PER	Program Event Recording
PERT	Program Evaluation and Review Techniques
PIOCS	Physical Input Output Control System
PKZIP	A data Compression Program
PL/1	Programming Language 1
PL/I	Programming Language 1
PLT	Program List Table
POWER	Priority Output Writers, Execution processor, and input Readers

PPFA	Page Printer Formatting Aid
PPT	Processing Program Table
PR/SM	Processor Resource/Systems Manager
PROC	PROCedure
PROP	PRogrammable OPerator
PS	Personal System
PSB	Program Specification Block
PSF	Print Service Facility
PSF/6000	Print Service Facility/6000
PTF	Program Temporary Fix
PW	PassWord
QISAM	Queued Indexed Sequential Access Method
QMF	Query Management Facility
QSAM	Queued Sequential Access Method
R/O	Read/Only
R/W	Read/Write
RACF	Resource Access Control Facility
RAMAC	Brand name and trademark of IBM DASD family
RAS	Reliability, Availability, Serviceability
RBA	Relative Byte Address
RC	Reason Code
RCB	Request Control Block
RDO	Resource Definition On-line
RECFM	RECord ForMat
RES	RESident
RETAIN	Remote Technical Assistance Information Network
REXX	REstructured eXtended eXecutor language
RJE	Remote Job Entry
RLS	Record Level Sharing
RMDS	Remote Management and Distribution System
RMF	Resource Measurement Facility
RPC	Remote Procedure Call
RPG	Report Program Generator
RPG II	A commercially Oriented Programming Language
RPL	Request Parameter List
RPM	Remote Print Manager
RRDS	Relative Record Data Set
RSCS	Remote Spooling Communications Subsystem
RSU	Recommended Service Upgrade
RTE	Remote Terminal Emulator
RU	Replaceable Unit
S/360	System/360
S/390	System/390
SAA	Systems Application Architecture
SAE	Strategic Applications Enabling
SAF	System Authorization Facility
SAM	Segmented Access Method
SCD	System Contents Directory
SCLM	Software Configuration & Library Manager
SCS	Spooling Communications System
SDF	Screen Definition Facility
SDF/CICS	Screen Definition Facility/Customer Information Control System
SDLC	Synchronous Data Link Control
SDSF	System Display and Search Facility
SGML	Standard Generalized Mark-up Language
SIE	Software Installation Express
SIS	Sequential Insert Strategy
SIT	System Initialization Table
SLI	Source Library Inclusion
SLIP	Serial Line Internet Protocol
SMF	System Management Facility
SMP	System Modification Program
SMP/E	System Modification Program/Extended
SMS	Stores Management System
SMTP	Spooling Communications System
SNA	Systems Network Architecture
SNI	SNA Network Interconnect
SP	System Product

SPG	Service Planning Guide
SPI	System Programming Interface
SPIE	Specify Program Interruption Exit
SPOOL	Simultaneous Peripheral Operation On-Line
SPUFI	SQL Processor Using File Input
SQL	Structured Query Language
SQL/DS	Structured Query Language/Data System
SQLCA	SQL Communication Area
SRL	Systems Reference Library
SRM	Systems Resources Manager
SSI	SubSystem Interface
SSP	System Services Program
STAE	Specify Task Abnormal Exit
STC	Started Task Control
SVA	Shared Virtual Area
SVC	SuperVisor Call instruction
SVS	Single Virtual Storage
SYSADM	SYStem ADMinistrator
SYSDEF	SYStem DEFinition
SYSIN	SYStem INput stream
SYSLOG	SYStem LOG
SYSOUT	SYStem OUTput stream
SYSREC	SYStem RECorder file
SYSRES	SYStem RESidence file
TCAM	TeleCommunications Access Method
TCB	Task Control Block
TCP/IP	Transmission Control Protocol/Internet Protocol
TCT	Terminal Control Table
TECB	Timer Event Control Block
TM	TradeMark
TME	Tivoli Management Environment
TMM	Tape Management Methodology
TOD	Time Of Day
TOR	Terminal Owning Region
TOS	Tape Operating System
TP	TeleProcessing
TPF	Transaction Processing Facility
TRS	Time Recording System
TSO	Time Sharing Option
TSO/E	Time Sharing Option Extensions
TSO/VTAM	Time Sharing Option/Virtual Telecommunications Access Method
UACC	User ACCess
UCB	Universal Character Buffer
UCS	Universal Character Set
UCS2	ISO 10646 16-bit Character Encoding Standard
UPSI	Use Program Switch Indicator
URL	Uniform Resource Locator
USASCII	Deprecated Term for ASCII
VAE	Virtual Addressability Extension
VM	Virtual Machine
VM/CMS	Virtual Machine/Conversational Monitor System
VM/ESA	Virtual Machine/Enterprise Systems Architecture
VMA	Virtual Machine Assist
VMPRF	VM Performance Reporting Facility
VS	Virtual Storage
VSAM	Virtual Storage Access Method
VSCR	Virtual Storage Constraint Relief
VSE	Virtual Storage Extended
VSE/ESA	Virtual Storage Extended/Enterprise Systems Architecture
VSE/ICCF	Virtual Storage Extended/Interactive Computing and Control Facility
VSE/POWER	Virtual Storage Extended/Priority Output Writers, Execution Peocessors
VSE/SP	Virtual Storage Extended/System Package
VSE/VSAM	Virtual Storage Extended/Virtual Storage Access Method

VTAM	Virtual Telecommunications Access Method
VTOC	Volume Table of Contents
VVDS	VSAM Volume Data Set
VVR	VSAM Volume Record
WS	Work Station
WSC	Washington Systems Center
WTM	Write Tape Mark
WTO	Write To Operator
WTOR	Write To Operator with Reply
WWW	World Wide Web
XCF	Cross-system Coupling Facility
XCTL	Transfer ConTroL
XEDIT	eXtended EDITor
XMT	Transmit
XRF	eXtended Recovery Facility

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

ITSO Redbook Evaluation

VSE to OS/390 Migration Workbook
SG24-2043-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

_____

SG24-2043-00
Printed in the U.S.A.

Artwork Definitions

Table Definitions

R2	REDB$EVA	621	621
R1	REDB$EVA	621	621, 621

Figures

id	File	Page	References
VAE	2043CH01	6	1 6
VAE4	2043CH01	7	2 7
ESA	2043CH01	8	3 8
OS3	2043CH01	9	4 9
MGTEAM	2043CH03	45	5 45
JMPVMC	2043CH03	49	6 49
F011	2043CH05	107	7 105, 106
WSC9741	2043CH05	113	8 117
F042	2043CH05	116	9
F054	2043CH06	136	11 136
CICSDMS	2043CH06	139	12 138
CICSLSC	2043CH06	146	13 145
CICSDS	2043CH06	146	14 146
F066	2043CH08	177	16
F2043A1	2043CH09	187	17 186, 186
DOSOPT	2043CH12	261	19 260
VS2OPT	2043CH12	262	20 261
VS2NAV	2043CH12	263	21 262
RESWDD	2043CH12	264	22 263
RESWDU	2043CH12	264	23 264
RESWDC	2043CH12	264	24 264
RESWD2	2043CH12	265	25 265
NOOO	2043CH12	265	26 262
F107	2043CH13	270	27 269
F108	2043CH13	271	28 269
F109	2043CH13		

		274	29	273
F1010	2043CH13	279	30	278
F1011	2043CH13	295	31	294
F1012	2043CH13	295	32	294
F1038	2043CH13	296	33	294
F1013	2043CH13	297	34	296
F1014	2043CH13	302	35	301
F1015	2043CH13	303	36	301
F1016	2043CH13	304	37	303
F1017	2043CH13	310	38	309
F1018	2043CH13	311	39	309
F1019	2043CH13	312	40	311
F1020	2043CH13	313	41	312
F1021	2043CH13	317	42	316, 324
F1022	2043CH13	318	43	317
F1032	2043CH13	318	44	318, 325
F1031	2043CH13	319	45	318
F1023	2043CH13	319	46	318, 319
F1024	2043CH13	320	47	318, 319, 324, 327
F1026	2043CH13	320	48	318, 320
F1027	2043CH13	321	49	318, 320, 324
F1030	2043CH13	324	50	318, 324, 324
F1025	2043CH13	325	51	318, 325
F1033	2043CH13	326	52	326
F1035	2043CH13	327	53	327
F1036	2043CH13	328	54	328
F1034	2043CH13	328	55	
CS45	2043CH17	366	56	365
JMACP	2043CH32	491	57	491

Headings

Index Entries

List Items

id	File	Page	References
TSK1	2043CH02		
		26	1 26
TSK2	2043CH02		
		26	2 26
TSK3	2043CH02		
		26	3 26
TSK4	2043CH02		
		26	4 26
TSK5	2043CH02		
		27	5 26
TSK6	2043CH02		
		27	6 26
TSK7	2043CH02		
		27	7 26
TSK8	2043CH02		
		27	8 26
TSK9	2043CH02		
		27	9 26
TSK10	2043CH02		
		27	10 26
TSK11	2043CH02		
		27	11 26
JANDS	2043CH04		
		71	4 78

Footnotes

id	File	Page	References
FNR3	2043CH10		
		222	6 222, 222, 222
FNMICRO	2043CH11		
		235	7 235

Tables

id	File	Page	References
CCHART	2043CH02		
		16	1
WHOAFF	2043CH02		
		26	2
NINMAIN	2043CH03		
		54	3
CNVMAIN	2043CH03		
		54	4
XYZAFF	2043CH03		
		55	5
OPRAFF	2043CH03		
		55	6
VSEJCL	2043CH04		

				358
OPTRC1	2043CH17			
		363	42	
				363
OPTRC4	2043CH17			
		363	43	
				363
CICOPT	2043CH17			
		367	44	
				367
TABDLY	2043CH25			
		403	45	
				432
VENPS	2043AX02			
		539	46	

Schedules

id	File	Page	References
JMPFSUM	2043CH03		
		56	
			56
JMPFDET	2043CH03		
		60	
			58

Processing Options

Runtime values:

- Document fileid ..... SG242043 SCRIPT
- Document type ..... USERDOC
- Document style ..... REDBOOK
- Profile EDFPRF40
- Service Level 0022
- SCRIPT/VS Release 4.0.0
- Date 98.10.20
- Time 04:41:52
- Device 3820A
- Number of Passes 4
- Index ..... YES
- SYSVAR D ..... YES
- SYSVAR G ..... INLINE
- SYSVAR X ..... YES

Formatting values used:

- Annotation ..... NO
- Cross reference listing ..... YES
- Cross reference head prefix only ..... NO
- Dialog ..... LABEL
- Duplex ..... YES
- DVCF conditions file ..... (none)
- DVCF value 1 ..... (none)
- DVCF value 2 ..... (none)
- DVCF value 3 ..... (none)
- DVCF value 4 ..... (none)
- DVCF value 5 ..... (none)
- DVCF value 6 ..... (none)
- DVCF value 7 ..... (none)
- DVCF value 8 ..... (none)
- DVCF value 9 ..... (none)
- Explode ..... NO
- Figure list on new page ..... YES
- Figure/table number separation ..... YES
- Folio-by-chapter ..... NO
- Head 0 body text ..... Part
- Head 1 body text ..... Chapter
- Head 1 appendix text ..... Appendix
- Hyphenation ..... NO
- Justification ..... NO
- Language ..... ENGL
- Keyboard 395
- Layout ..... OFF
- Leader dots ..... YES
- Master index ..... (none)
- Partial TOC (maximum level) 4
- Partial TOC (new page after) ..... INLINE
- Print example id's ..... NO

Print cross reference page numbers ..... YES
Process value ..... (none)
Punctuation move characters ..... .,
Read cross-reference file ..... (none)
Running heading/footing rule ..... NONE
Show index entries ..... NO
Table of Contents (maximum level) 3
Table list on new page ..... YES
Title page (draft) alignment ..... RIGHT
Write cross-reference file ..... (none)

Imbed Trace

Page 0	2043SU
Page 0	2043VARS
Page 0	REDB$POK
Page i	REDB$ED1
Page i	2043EDNO
Page i	REDB$ED2
Page xxi	2043ABST
Page xxi	2043ACKS
Page xxii	REDB$COM
Page xxii	2043IMBD
Page 1	2043CH01
Page 12	2043CH02
Page 40	2043CH03
Page 67	2043CH04
Page 95	2043CH05
Page 132	2043CH06
Page 154	2043CH07
Page 167	2043CH08
Page 183	2043CH09
Page 206	2043CH10
Page 234	2043CH11
Page 247	2043CH12
Page 265	2043CH13
Page 328	2043CH14
Page 331	2043CH15
Page 347	2043CH16
Page 349	2043CH17
Page 367	2043CH18
Page 373	2043CH19
Page 380	2043CH20
Page 385	2043CH21
Page 388	2043CH22
Page 391	2043CH23
Page 395	2043CH24
Page 399	2043CH25
Page 417	2043CH26
Page 435	2043CH27
Page 441	2043CH28
Page 454	2043CH29
Page 456	2043CH30
Page 472	2043CH31
Page 479	2043CH32
Page 518	2043CH33
Page 527	2043CH34
Page 533	2043AX01
Page 537	2043AX02
Page 541	2043AX03
Page 553	2043SPEC
Page 553	REDB$SPE
Page 553	2043TMKS
Page 556	2043BIBL
Page 559	REDB$BIB
Page 560	REDB$ORD
Page 563	2043GLOS
Page 582	2043ABRV
Page 620	REDB$EVA