IMS/ESA

Administration Guide: System

Version 6

SC26-8730-04

IMS/ESA

Administration Guide: System

Version 6

SC26-8730-04

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

Fifth Edition (October 1999)(Softcopy Only)

This edition replaces and makes obsolete the previous edition, SC26-8730-03. This edition is available in softcopy only. The technical changes for this edition are summarized under “Summary of Changes” on page xv and are indicated by a vertical bar to the left of a change.

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

A form for readers’ comments is provided at the back of this publication. If the form has been removed, address your comments to:

- IBM Corporation, BWE/H3
- P.O. Box 49023
- San Jose, CA, 95161-9023
- U.S.A.

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

Contents

Notices

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 that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
555 Bailey Avenue, W92/H3
P.O. Box 49023
San Jose, CA 95161-9023

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Function Printing
AFP
BookManager
CICS
DATABASE 2
DataPropagator Nonrelational
DPropNR
DB2
DFSMS
ES/9000
IBM
IMS
IMS/ESA
IMS Client Server/2
MVS
MVS/DFP
MVS/ESA
OS/390
PSF
RACF
Resource Measurement Facility
RMF
SP
VTAM

Product Names

In this book, the licensed program “DB2 for MVS/ESA” is referred to as “DB2”.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Preface

This book is a guide to administering IMS/ESA (hereafter called IMS), which can consist of the IMS Database Manager, or the IMS Transaction Manager, or both.

This book consists of the following chapters:

- "Chapter 1. Introduction" on page 9 describes the IMS environments and presents an overview of system administration.
- "Chapter 2. Documenting Your IMS System" on page 35 explains how to extract IMS requirements, participate in design reviews, establish naming conventions, and document an IMS system.
- "Chapter 3. Defining Your System" on page 43 identifies the system definition macros and their required and optional control parameters.
- "Chapter 4. Establishing IMS Security" on page 115 provides information to help you establish resource security for the IMS online system.
- "Chapter 5. Designing a System with Extended Recovery Facility" on page 159 introduces the Extended Recovery Facility (XRF) and describes system administration activities when an online IMS subsystem is defined to work with an alternate IMS subsystem.
- "Chapter 6. Testing Your System" on page 237 describes the two phases of testing related to system administration: testing application programs prior to cutover, and testing changes or corrections to application programs after their initial use.
- "Chapter 7. Monitoring Your System" on page 247 explains how to establish monitoring procedures.
- "Chapter 8. Tuning Your System" on page 265 presents a general strategy for tuning an IMS online system.
- "Chapter 9. Modifying Your System Design" on page 295 summarizes the planning activities needed when changes are necessary to an IMS system design.
- "Chapter 10. Administering a Data Sharing Environment" on page 305 describes system administration activities when you allow more than one IMS online or batch system concurrent access to a database.
- "Chapter 11. Administering the IMS Spool API" on page 337 provides design and operational advice for the use of the IMS Data Language/I (DL/I) Spool application program interface (API).
- "Chapter 12. Administering the Remote Site Recovery Complex" on page 347 introduces Remote Site Recovery (RSR) and describes quick recovery from an interruption of computer services at an active (primary) site.

A bibliography lists non-IMS publications cited in this book.

Prerequisite Knowledge

IBM offers a wide variety of classroom and self-study courses to help you learn IMS. For a complete list, see the IMS home page on the World Wide Web at: `http://www.software.ibm.com/data/ims`

Before using this book, you should have a good understanding of the IMS environment (DB/DC, DBCTL, or DCCTL) that you are managing. You should also have a basic understanding of database processing and the access methods used

by DL/I. It is helpful to know the purpose of the different types of DL/I calls, the IMS application program structure, and the tasks associated with application program design.

For an introduction to database applications and basic IMS terminology, read *IMS/ESA General Information*, which is on the Web at the above address. For information on this release of IMS, see *IMS/ESA Release Planning Guide*. Establishing operating procedures presumes an understanding of IMS commands. This information is in *IMS/ESA Operations Guide*.

How to Use This Book

Use this book in conjunction with *IMS/ESA Installation Volume 1: Installation and Verification* and *IMS/ESA Installation Volume 2: System Definition and Tailoring* to help coordinate the design, installation, definition, and modification of an IMS system. See *IMS/ESA Installation Volume 1: Installation and Verification* and *IMS/ESA Installation Volume 2: System Definition and Tailoring* when you need details for installing an IMS system. See *IMS/ESA Installation Volume 2: System Definition and Tailoring* when you need details for designing, defining, or modifying an IMS system.

You can also use this book as an information source for monitoring and tuning IMS; it explains the use of the IMS Monitor and other utilities.

This book describes the environments available with the IMS Database Manager and with the IMS Transaction Manager. Some sections of this book apply only to specific environments. A table at the beginning of each of these sections denotes the applicable environments for that section. No environments table appears in sections that apply to all environments. The specific environments are:

- Systems using both the Database Manager and the Transaction Manager (DB/DC subsystems)
- Systems using the database control (DBCTL) subsystem
- Systems using the data communications control (DCCTL) subsystem

Change Indicators

Technical changes are indicated in this publication by a vertical bar (|) to the left of the changed text. These changes were added to IMS Version 6 by new product line items.

Summary of Changes

Changes to The Current Edition of This Book for V6

This edition includes technical and editorial changes.

Changes to This Book for V6

This book contains new and changed information about the following enhancements:

- APSB SAF Security
- DEDB Online Change
- Distributed Sync Point
- Shared VSO
- Fast DB Recovery
- OSAM Database Coupling Facility Caching
- Shared Queues and Shared EMH
- Shared SDEPs

Changes have been made to the following chapters:

Chapter 1 Introduction

Chapter 2 Documenting Your IMS System

Chapter 3 Defining Your System

Chapter 4 Establishing IMS Security

Chapter 5 Designing a System with Extended Recovery Facility

Chapter 7 Monitoring Your System

Chapter 8 Tuning Your System

Chapter 10 Administering a Data Sharing Environment

Chapter 12 Administering the RSR Complex

This edition incorporates new technical information for Version 6, as well as editorial changes and technical corrections made to previously published information.

Library Changes for Version 6

The IMS/ESA Version 6 library differs from the IMS/ESA Version 5 library in these major respects:

- *IMS/ESA Common Queue Server Guide and Reference*

  This new book describes the IMS Common Queue Server (CQS).
- *IMS/ESA DBRC Guide and Reference*

  This new book describes all the functions of IMS Database Recovery Control (DBRC).
- The IMS Application Programming summary books (*IMS/ESA Application Programming: Database Manager Summary*, *IMS/ESA Application Programming: Transaction Manager Summary*, and *IMS/ESA Application Programming: EXEC DLI Commands for CICS and IMS Summary*) are no longer included with the IMS library.

- The Softcopy Master Index is not included.
- All information about IRLM 1.5 and data sharing using IRLM 1.5 has been removed from the IMS V6 books. If you use IRLM 1.5, and want to migrate to using IRLM 2.1 and Sysplex data sharing, see *IMS/ESA Release Planning Guide*.
- The chapter that was titled ″Database Control (DBCTL) Interface″ in the *IMS/ESA Customization Guide* has been revised for Open Database Access (ODBA) and moved to ″Appendix A, Using the Database Resource Adapter (DRA)″ in the *IMS/ESA Application Programming: Database Manager*.

Part 1. System Planning and Definition

Chapter 1. Introduction

This chapter outlines the administrative activities for an IMS system. It also introduces the concepts that are central to administering an IMS system.

In this Chapter:

Planning for Administrative Activities

The areas of responsibility associated with administering online IMS systems are:

- Designing an IMS online system
- Establishing operating procedures that meet application requirements
- Maintaining a production system that is responsive to end users
- Integrating new applications or major design changes into the current system

To meet these responsibilities, you must coordinate many activities that occur during an application development cycle. Performance of these activities results in:

- Documentation of the IMS network
- Specifications for IMS system definition and execution control parameters
- Procedural controls for operation
- Strategies for monitoring and audit trails

After entry into production mode, your ongoing activities support:

- Auditing operations and end user service
- Monitoring and gathering production statistics
- Establishing procedures to control changes to the online system design
- Testing the online system after application and IMS changes

Figure 1 on page 11 is an overview of the administration activities. The activities in the first column of the figure take place during the design phase, the second column's activities occur during development of application code, and those in the third column occur during test. The vertical center line marks the transition to production mode.

Before the start of production mode, administration activities lead to two major activities:

- Generating the system and preparing JCL (job control language)
- Developing operations procedures

The items to the right of the center line in Figure 1 on page 11 reflect the ongoing system administration activities. The column headed by `PRODUCTION` emphasizes awareness of the day-to-day operation and performance of the system. The next column's activities are concerned with maintenance. Minor application design changes, problem resolution, and any IMS maintenance are included in this category. The column on the far right identifies the activities needed to integrate additional applications or implement an application package. For a major addition, activities are similar to those on the left-hand side of the figure. Other design

changes that do not involve terminal or network modifications can be handled without shutting down the IMS system. For these simple design changes, the associated activities lead to a revision of operating procedures in maintenance mode.

After startup, revisions of IMS system definition and operating procedures become key activities. In this context, you might decide to change applications during online operation. The interpretation of system performance then becomes an important activity, supported by monitoring and performance analysis.

Each row in Figure 1 on page 11 represents a set of activities, each having its own characteristics:

- Analyzing user requirements involves examining the application documentation for the IMS function required and the expected workload.
- Collecting online requirements concerns specifications for IMS system definition, system data set allocation, and initial JCL.
- Preparing the IMS network involves interaction with system and network generation activities.
- Establishing security procedures encompasses the design and implementation of a security strategy.
- Developing an operations plan produces operations control documents and provides for audit of the control of production cycles.
- Forming a monitoring strategy results in monitoring the system and gathering performance data.
- Establishing criteria for performance leads to performance analysis and tuning activities.

Related Reading: This publication does not address the detailed planning needed to establish operating procedures. For information on establishing operating procedures, see *IMS/ESA Operations Guide*.

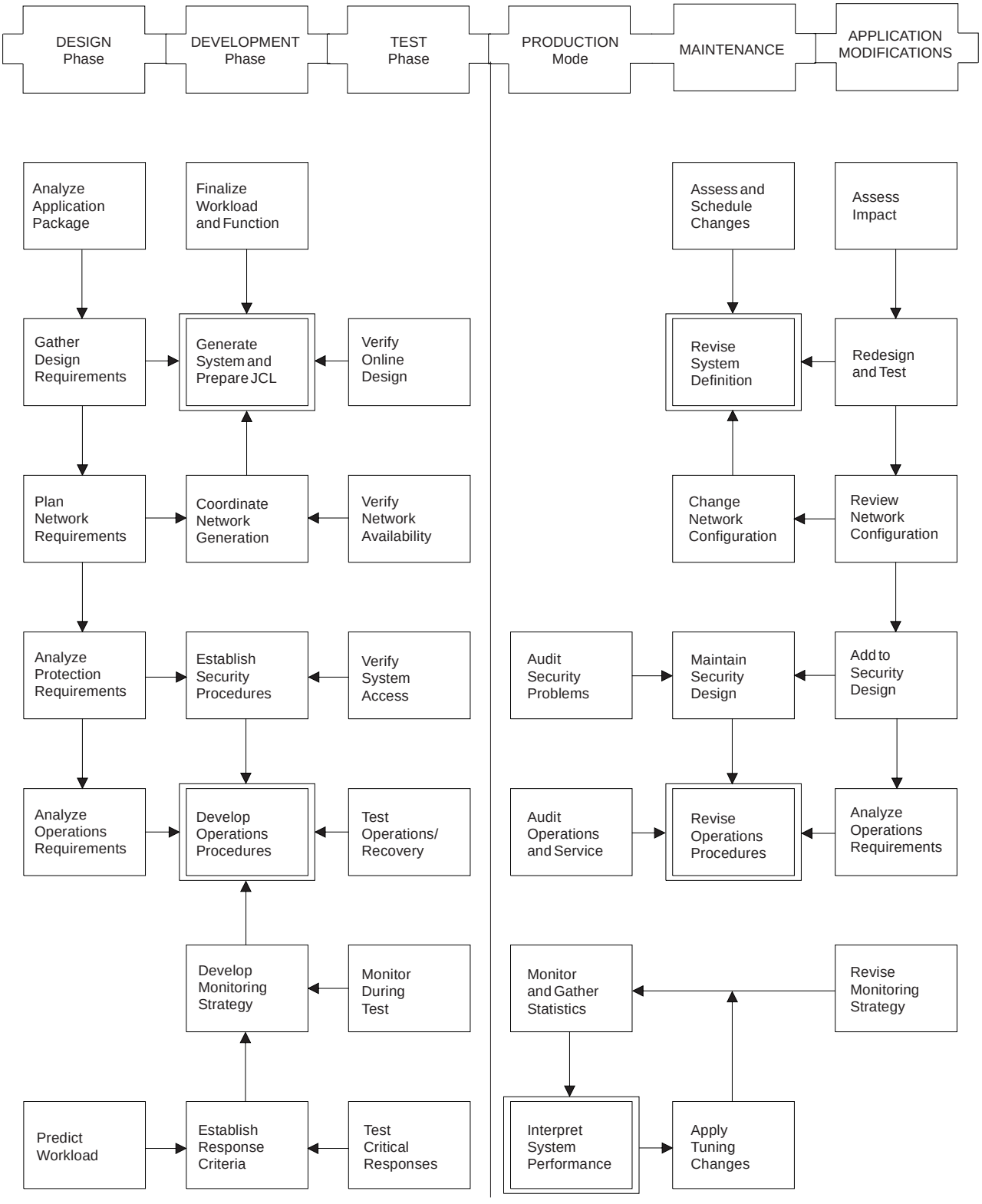


Figure 1. Chart of IMS System Administration Activities

The IMS Environments

IMS consists of two licensed programs: the IMS Database Manager (DB) and the IMS Transaction Manager (TM). With the Database Manager, you can generate the batch environment and the database control (DBCTL) environment. With both the Database Manager and the Transaction Manager, you can generate the DB/DC environment. With the Transaction Manager, you can generate the data communication control (DCCTL) environment. [1]

Each of the IMS environments is a distinct combination of hardware and programs that supports distinct processing goals. The environments and the goals they support are shown in Table 1.

Table 1. IMS Environments

Environments	Data Processing Goals
DBCTL (See "The DBCTL Environment" on page 17)	• Process network transactions without the Transaction Manager—that is, use the Database Manager with a transaction management subsystem (for example, CICS). • Run batch application programs using DB batch at certain intervals (for example, process a payroll or produce an inventory report). • Run database utilities using DB batch.
DB/DC (See "The DB/DC Environment")	• Enable terminal users to retrieve data and modify the database with satisfactory real-time performance. (Some typical applications are banking, airline reservations, and sales orders.) • Ensure that retrieved data is current. • Distribute transaction processing among multiple processors in a communications network. • Run batch application programs using DB batch at certain intervals (for example, process a payroll or produce an inventory report). • Run database utilities using DB batch.
DCCTL (See "The DCCTL Environment" on page 19)	• Process network transactions without the Database Manager by using the Transaction Manager with an external database management subsystem. • Maintain system log information for restart by using DBRC. • Run batch application programs in a TM batch region by using the Transaction Manager to do batch processing with DATABASE 2 (DB2).

The following sections describe the hardware and programs for each environment. They also explain the requirements, options, and limitations that are characteristic of each environment.

The DB/DC Environment

In the DB/DC environment, data is centrally managed for applications that are being executed concurrently and made available to terminal users. Database Recovery Control (DBRC) facilities help to manage database availability, data sharing, and system logging.

The basic unit of work is the transaction. Transaction processing consists of:

1. Data sharing and the Extended Recovery Facility (XRF) are often considered environments, but they are really special cases of the three environments listed here.

- Receiving a request for work that has been entered at a terminal. The request is in the form of a transaction code, which identifies the kind of work to be performed and the data needed to do it.
- Invoking a program to do the work, and preparing a response for the terminal operator (for example, an acknowledgment of work performed or an answer to an inquiry).
- Transmitting the response to the terminal that requested the work.

The simplest kind of transaction involves two messages: an input message from the terminal user and an output message in return. Application programs can also send messages to terminals other than the input source, and they can generate transactions.

Figure 2 represents a DB/DC environment.

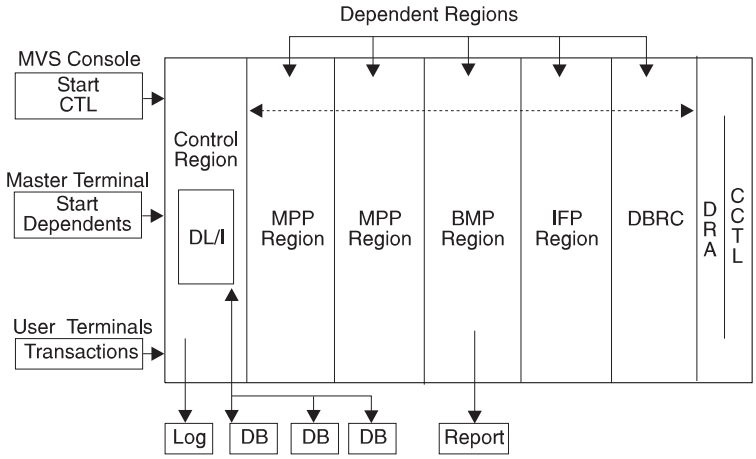


Figure 2. Example of a DB/DC Environment

Control Region

The address space holding the control program that runs continuously in this environment. It is normally started by using the MVS `START` command. The control region then automatically initiates the DBRC address space.

In Figure 2, DL/I is shown as part of the control region, but it does not need to run there. You can run DL/I in its own address space.

Dependent Region

The other address spaces containing the application programs that process the transactions sent from user terminals. The dependent regions are initiated by an MVS `START` command or by a `/START REGION` command from the IMS master terminal.

MPP Region	Message Processing Program region used for processing messages.
BMP Region	Batch Message Processing region used for processing batch operations.
IFP Region	IMS Fast Path region used for processing Fast Path messages.
DBRC Region	Database Recovery Control region used for controlling the logs and recovering the databases. It also controls the data sharing

environment by allowing (or preventing) access to databases by various IMS subsystems sharing those databases.

The Master Terminal

The master terminal is the control center of the DB/DC environment. The Master Terminal Operator (MTO) must know all the operating aspects of the system and be familiar with the purpose and action of all the IMS commands that are entered. Some characteristics of the master terminal are:

- It is used to enter commands that start, stop, and restart the system.
- As a logical terminal, it receives system messages.
- The primary control of the network and connecting terminals is performed through the master terminal. It can start and stop communication lines and assign logical terminals to physical terminal destinations.
- The status of the system can be displayed from the master terminal. Such items as the number of transactions to be processed, the number of programs and databases that are active, and the status of communication lines can be requested.
- If a program or database error occurs, commands can be entered from the master terminal to prevent further processing against the affected resource and to prevent input or output activity for terminals. These recovery actions can also apply to a recovery of the entire system after an abnormal termination occurs.
- If an exception condition occurs, the status of the Online Log Data Set (OLDS) can be displayed, and the function of the OLDS can be controlled from the master terminal.

If the master terminal becomes inoperable, the operating system console can be used as a backup. The MTO can either operate the IMS system from the system console or assign the master terminal LTERM (logical terminal) to an alternate terminal. Messages continue to be routed to the old master terminal LTERM until the LTERM is assigned to the system console. [2] If the system console is used to continue operation, terminal security is identical to that of the master terminal.

IMS Commands

IMS commands are messages that are entered by the operator and sent to the control region. IMS commands are identified by a slash (/) as the first character. They consist of a verb, optional password, and keywords. Values can be entered with each keyword. This command, for example, might be used to start a message region:

```
/START(MTO005)   REGION   MSGREG1
verb  password   keyword  value
```

IMS commands can start, stop, add, change, delete, and display the status of resources. They can also assign relationships and values; for example, they can connect logical terminals to non-switched physical terminals, direct transactions to classes and classes to regions, and alter the priorities of transactions.

Related Reading: For more information on IMS commands, see *IMS/ESA Operator's Reference*.

Control Region

The IMS control region owns all the databases that can be accessed by online application programs and is responsible for all physical input/output to the

2. The address of the system console is LINE 1 PTERM 1.

databases. The control region holds the *control program*, which continuously runs in the control region and controls the processing in other regions. The control region services all DL/I calls. It supervises the processing of messages and all the communication traffic for the connected terminals. The control region also manages information for restart and recovery purposes and operates the IMS system log.

Fast DB Recovery Region

The Fast DB Recovery region is a separate IMS control region that monitors an IMS subsystem, detects failure, and recovers any database resources that are locked by the failed IMS, making them available for other IMS subsystems.

The Fast DB Recovery region is executed by the IMS SYSGEN-supplied cataloged procedure. It must be started after the IMS subsystem that it tracks is started.

To enable a DB/DC subsystem for Fast DB Recovery, you specify the FDRMBR parameter in the IMS procedure. The FDRMBR parameter defines the DB/DC system as Fast DB Recovery-capable.

MPP Regions

MPP regions are started either by the master terminal operator or by JCL if the control program is running. The control program schedules application programs within the MPP regions. The application programs then run, accessing the online databases and obtaining their transaction input from the message queues. The application programs cannot access MVS files or issue MVS checkpoints. The application program output messages can be directed to LTERMs or to other application programs. An application program can remain scheduled in an MPP region even when there is no work to process for that region. The MPP region remains in a wait-state (wait-for-input mode) until there is more work for the region to process.

BMP Regions

MVS schedules the BMP regions. The application programs in those regions are determined by the JCL used to start each region, not by the control region. These application programs can access databases owned by the control region and MVS data sets owned by their BMP regions. This includes data entry databases (DEDBs) and main storage databases (MSDBs).

Application programs in BMP regions can access input and output message queues; they can also execute in wait-for-input mode. To access the input message queues, you specify, in the JCL for a BMP region, a transaction code you want to access. Specifying this transaction code also gives you access to the output message queues using terminal program communications blocks (PCBs) in the application program's specification block (PSB). Even without access to input message queues, if you specify an output LTERM or transaction code in the JCL for the region, the application program can issue output messages.

IFP Regions

Two types of programs run in IFP regions:

- Application programs for processing Fast Path messages; these are called message-driven programs.
- Utilities that process DEDBs; these are BMPs.

Related Reading: For more information on the types of Fast Path processing and the administration of a DB/DC environment that includes Fast Path, see *IMS/ESA Administration Guide: Database Manager*.

Data Sharing

Data can be shared among dependent regions and with other IMS systems. The other systems can be DB/DC or DBCTL. If you intend to share data at the block level, the Internal Resource Lock Manager (IRLM) must be present in every environment that participates. IRLM runs in its own address space.

Running the Extended Recovery Facility

The Extended Recovery Facility (XRF) is a combination of programs that includes two DB/DC environments to provide a high level of IMS availability to end users. One environment is active and is called the active system. The other environment continuously tracks the processing of the first and is called the alternate system. The alternate system is ready to take over if the active system fails, or if a planned takeover is initiated (to perform maintenance, for example).

Related Reading:

- For more information on XRF, see "Chapter 5. Designing a System with Extended Recovery Facility" on page 159.
- For more information on Remote Site Recovery (RSR), see "Chapter 12. Administering the Remote Site Recovery Complex" on page 347.

DB Batch

DB batch consists of a batch region—one address space—where an application program and DL/I routines reside. The batch job that runs here is initiated with JCL, like any operating system job.

Figure 3 represents a batch environment in which the batch job is submitted via a TSO terminal, and an application program is run in order to read from an update file, write to a database, and produce a report. For example, an inventory application might be run—one that reads sales records (inventory reductions) and supply records (inventory increases), updates a database accordingly, and prints an inventory sales report.

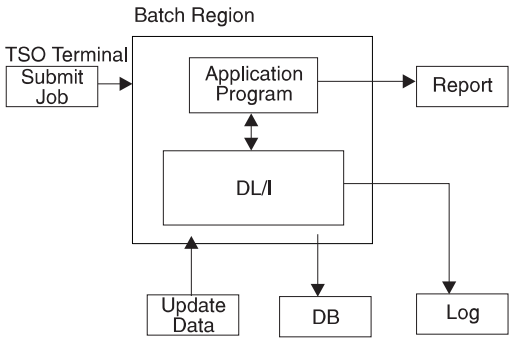


Figure 3. Example of a DB Batch Environment

The RSR Environment

The remote site recovery (RSR) environment allows you to recover quickly from an interruption of computer services at a primary site. IMS database and online transaction information is continuously transmitted to a secondary site. In the event of a service interruption at the active site, this secondary site is ready to take over the work from the active site.

Related Reading: For more information on RSR, see "Chapter 12. Administering the Remote Site Recovery Complex" on page 347.

The DBCTL Environment

The DBCTL environment is similar to the DB/DC environment; a DL/I region owns the databases to be processed. DL/I also exists in the DBCTL environment, although DL/I must run in its own address space. Database Recovery Control (DBRC) facilities, required for DBCTL, help to manage database availability, data sharing, system logging, and database recovery.

The greatest dissimilarity between DBCTL and DB/DC is that DBCTL does not support user terminals, a master terminal, or message handling. Therefore, no MPP regions exist. The BMP region is used only by batch applications and utilities. External program subsystems can, however, use an interface that does handle messages—a coordinator controller (CCTL). [3] The interface between the CCTL and the control region is the database resource adapter (DRA). The DRA resides in the same address space as the CCTL.

The CCTL handles message traffic and schedules application programs, all outside the DBCTL environment. It passes database calls through the interface to the control region, which sends the calls to DL/I and passes results back through the interface to the CCTL.

The information in this book that describes the IMS online system applies to both DB/DC and DBCTL. Exceptions are noted as not applicable to DBCTL.

Figure 4 shows an example of the DBCTL environment.

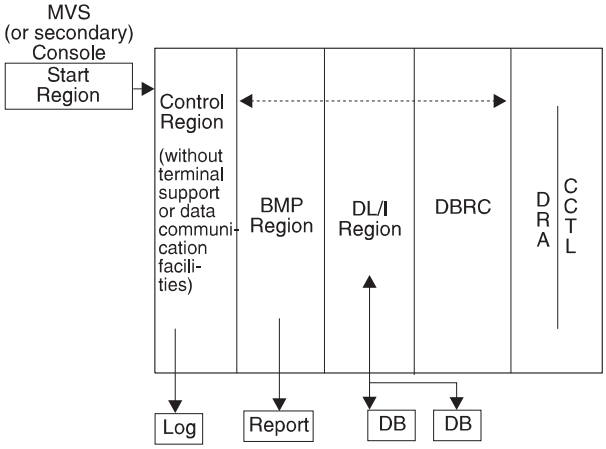


Figure 4. Example of a DBCTL Environment

Even though the DBCTL environment includes no master terminal, you can still control the environment with IMS commands. The commands and command functions that control message processing are not operable here, but the remainder are. They can be entered through the MVS console or a secondary console. The control region recognizes commands by their first character, a slash (/). You can choose a different first character during system definition, or as an execution parameter.

Note: References to the master terminal operator (MTO) in this book refer to either the DB/DC MTO or to the DBCTL operator.

3. The same interface exists in the DB/DC control region, so it is possible to use a CCTL with a DB/DC environment.

Output messages from a command are sent to the console that entered the command. You can also specify other consoles to receive unsolicited output. These consoles are those that fall into the category you define using the IMS-generating macro, IMSCTRL.

Databases Supported

The DBCTL environment supports all full-function databases (HSAM, SHSAM, HISAM, SHISAM, HDAM, and HIDAM).

BMP regions in a DBCTL environment can access GSAM databases. BMP regions can also access external subsystems (for example, DB2), because DBCTL supports the external subsystem interface.

Fast Path data entry databases (DEDBs) are also supported, but main storage databases (MSDBs) are not supported.

Utilities Supported

The following utilities can be run in BMP regions:

- Online Change utility
- Online Database Image Copy (OLIC) utility
- Batch Backout utility
- PSBGEN utility
- DBDGEN utility
- ACBGEN utility

Fast DB Recovery Region

The Fast DB Recovery region is a separate IMS control region that monitors an IMS subsystem, detects failure, and recovers any database resources that are locked by the failed IMS, making them available for other IMS subsystems.

The Fast DB Recovery region is executed by the IMS SYSGEN-supplied cataloged procedure. It must be started after the IMS subsystem that it tracks is started.

To enable a DBCTL subsystem for Fast DB Recovery, you specify the FDRMBR parameter in the DBC procedure. The FDRMBR parameter defines the DBCTL system as Fast DB Recovery-capable.

Data Sharing

As in the DB/DC environment, data can be shared between dependent regions and with other IMS systems. The other systems can be either DB/DC or DBCTL environments. If you intend to share data at the block level, IRLM must be present in every environment that participates.

Running an Alternate DBCTL Environment

You cannot run DBCTL environments with XRF, but you can still run two DBCTL environments—an active and an alternate—and thereby increase system availability. However, the alternate DBCTL environment does not track the processing of the active environment. The console operator must use the emergency restart command (`/ERESTART`) against the alternate system in order to make it the active environment.

DB Batch

DB batch consists of a batch region—one address space—where an application program and DL/I routines reside. The batch job that runs here is initiated with JCL, like any operating system job.

Figure 5 represents a batch environment in which the batch job is submitted via a TSO terminal, and an application program is run in order to read from an update file, write to a database, and produce a report. For example, an inventory application might be run—one that reads sales records (inventory reductions) and supply records (inventory increases), updates a database accordingly, and prints an inventory sales report.

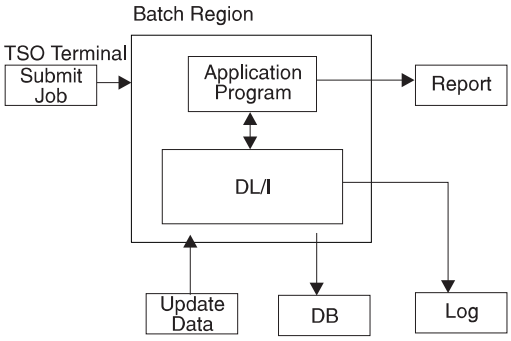


Figure 5. Example of a DB Batch Environment

The DCCTL Environment

DCCTL is a Transaction Manager subsystem that has no database components. A DCCTL environment is similar to the DB/DC environment. The primary difference is that a DCCTL control region owns no databases and does not service DL/I database calls.

DCCTL, in conjunction with the IMS External Subsystem (ESS) Attach Facility, provides a Transaction Manager facility to external subsystems (for example, DB2). In a DCCTL environment, transaction processing and terminal management is identical to transaction processing and terminal management in a DB/DC environment. DCCTL contains the programming support necessary for:

- Master terminal support
- Terminal network support
- Data communication
- Message handling
- Transaction processing
- Application program execution
- IMS command execution

DCCTL also supports online change, Message Format Service (MFS), Multiple Systems Coupling (MSC), and Database Recovery Control (DBRC).

DBRC is required, and is used to maintain system log information for restart. DBRC in a DCCTL environment maintains logs only for transactions. External database subsystems must maintain their own database logs.

DCCTL consists of three address space types:

- Control region
- DBRC
- Dependent regions (up to 255)

Dependent regions and DBRC are subordinate to the control region.

The DCCTL control region contains three structural components:

- A data communication manager, which controls terminal states and input/output message traffic. It also contains security controls that prevent unauthorized access to DC resources.
- A message manager, which is the read/write and I/O interface between terminal input from the data communication manager and the scheduling services of the Transaction Manager.
- A Transaction Manager, which manages MPPs, BMPs, and IFPs, schedules application programs in those dependent regions, and owns and responds to the application programming interface (API).

Each manager (data communication manager, Transaction Manager, and message manager) controls the use of its resources and the recoverability of its resources during a system failure. Like DB/DC dependent regions, MPP, BMP, and IFP dependent regions are used by the Transaction Manager to schedule application programs.

Figure 6 represents a DCCTL environment that is attached to an external subsystem.

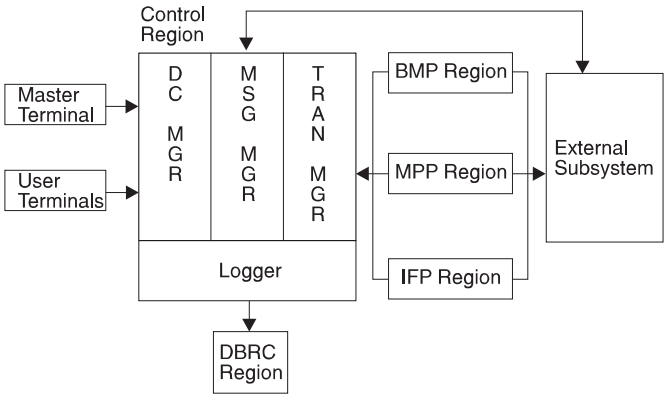


Figure 6. Example of a DCCTL Environment and Attached Subsystem

DCCTL coordinates the sync point recovery process with the connected external subsystems. DCCTL ensures that database updates and terminal messages are committed when an application program reaches a sync point.

Two methods exist for connecting a DCCTL control region to another subsystem. You can have DCCTL use the control region EXEC parameter, SSM, to select the PROCLIB member. Or, you can use the `/START SUBSYSTEM SSM` command, which allows DCCTL to SUBSYSTEM SSM command connect to other subsystems even though you did not request this option when you started IMS.

You can also specify the dependent region EXEC parameter, SSM, for dependent regions. The control region SSM member definition allows the dependent region to select one or more external subsystem connections. The SSM member can contain no definitions (null members) to prevent a connection to an external subsystem.

Related Reading: For more information on how to use the ESS interface, see *IMS/ESA Customization Guide*.

After you use the `/START SUBSYSTEM SSM` command, you must stop active dependent regions and then restart them if they require an external subsystem connection.

Related Reading: For more information on using the `/START SUBSYSTEM SSM` command, see *IMS/ESA Operator's Reference*.

After IMS has established a connection between the dependent region and the external subsystem, a thread is created between the connected regions. The thread is used for subsequent application program calls, committing the data, or in failure situations, backing out the data.

The application programs managed by DCCTL are identical to those managed by the DC manager and TM manager in the DB/DC environment.

Databases Supported

DCCTL is a compatible communications front end for accessing DB2 and GSAM databases.

With GSAM databases, DCCTL uses sequential, non-IMS data sets with a BMP. Application programs can also issue symbolic checkpoint (CHKP) and extended restart (XRST) calls against a GSAM data set using the I/O PCB. The ability to issue CHKP and XRST calls allows data set repositioning.

DCCTL accesses DB2 databases through the External Subsystem (ESS) Attach Facility. The control region initiates contact with other subsystems. The other subsystems that DCCTL can access are defined in an IMS.PROCLIB member. You need to provide an IMS.PROCLIB member for DCCTL defining all other subsystems that can be accessed by DCCTL. The subsystem definition contains the information DCCTL uses to communicate with the other subsystem.

Related Reading: For more information on the ESS Attach Facility and on defining IMS.PROCLIB members, see *IMS/ESA Customization Guide*.

DCCTL handles transaction management for online IMS applications that need to access external subsystems.

Commands Supported

All IMS commands are supported in a DCCTL environment except database commands and database-related keywords.

Related Reading: For a complete listing of valid DCCTL commands and keywords, see *IMS/ESA Operator's Reference*.

Utilities Supported

These utilities can be run in DCCTL regions:

- Online Change utility
- Log Archive utility
- Log Recovery utility
- PSBGEN utility
- DBDGEN utility

TM Batch

IMS TM supports a batch region for running application programs. IMS applications cannot use the ESS Attach Facility to issue SQL calls in batch. This support is provided by an external subsystem.

Related Reading: If your external subsystem is DB2, see *DATABASE 2 Application Programming Guide* for a description of the steps required to allow batch programs to issue SQL calls.

You can connect DB2 in an IMS TM batch environment in one of two ways. You can use the SSM parameter on the TM batch-region execution JCL and specify the actual name of the batch program on the MBR parameter. Alternatively, you can code the DDITV02 DD statement on the batch-region execution JCL and specify the name of the DB2 module, DSNMTV01, on the MBR parameter.

Related Reading: For additional options or requirements, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Figure 7 shows TM batch with an attached external subsystem.

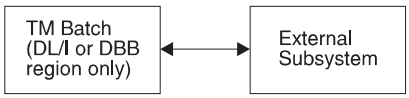


Figure 7. Example of TM Batch and Attached Subsystem

Valid TM batch region types are DBB, DLI, or UPB. All other region types are not applicable to the TM batch environment.

You specify generated program specification blocks (GPSBs) for a TM batch environment using the PSB parameter in the DBBBATCH and DLIBATCH procedures.

DCCTL Resembles IMS

IMS and DCCTL are very similar, except that no database components exist in DCCTL. Similarities between IMS and DCCTL include:

- The control region and dependent region initialization and termination for DCCTL are identical to IMS initialization and termination.
- The system definition process for DCCTL is identical to the process required for defining and generating an IMS system.
- The restart process for DCCTL is the same as for IMS.
- The stage 1 input for an IMS system can be used for a stage 1 input for DCCTL without removing the database definitions. [4]
- Because DCCTL is a subset of an IMS system, all techniques used to diagnose errors in IMS are valid for DCCTL.

Data Communication Calls: The following data communication calls are available to application programs in a DCCTL environment:

AUTH	GU
CHNG	ISRT
CMD	PURG
GCMD	SETO
GN	

System Service Calls: The following system service calls are available to application programs in a DCCTL environment:

APSB	ROLL
CHKP	ROLS

4. You will need to make some changes to define a DCCTL system. See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for more information about analyzing macros for system definition.

DPSB	SETS
INIT	SETU
INQY	SYNC
LOG	XRST
ROLB	

Status AD: Application program calls passed to DCCTL receive an AD status code if the call function is not supported or if a database PCB is passed as part of the call list.

Related Reading: For more information on application programming in a DCCTL environment, see *IMS/ESA Application Programming: Transaction Manager*.

Exit Routines: You do not need to change your IMS exit routines or your current IMS application programs that access other subsystem resources in a DCCTL environment. However, application programs that contain a mixture of calls that access other subsystems and IMS databases require changes. All DL/I calls that use a database PCB receive status code AD.

Automated Operator Transactions: Automated operator transactions are started the same way as IMS transactions are started. Automated operator transactions run as IMS application programs with the authority to issue a subset of DCCTL commands using DL/I calls.

Concepts for the System Administrator

The remainder of this chapter presents concepts that are central to controlling the resources of an IMS system. The remaining chapters of this book assume a thorough understanding of these concepts.

Dynamic Allocation with IMS

If data sets that belong to databases (or an IMS Monitor data set residing on a tape device) are specified with JCL in a control region procedure, they are initially allocated when the control region starts up. You can specify that these database data sets be dynamically allocated when needed and deallocated when no longer in use.

Using the IMS macro DFSMDA (see *IMS/ESA Utilities Reference: System*), you declare those data sets that are subject to dynamic allocation and deallocation.

- Database data sets can be dynamically allocated explicitly with the `/START` command or implicitly when an application program is scheduled. Database data sets can be deallocated with the `/DBRECOVERY` command.
- For DEDB area data sets, an implicit allocation occurs at first access by an application program; the `/STOP` command also deallocates the data set.
- An IMS Monitor data set can be dynamically allocated at the time the IMS Monitor is started with the `/TRACE SET ON` command and deallocated by the `/TRACE SET OFF` command.
- RECON data sets, online log data sets (OLDSs), write-ahead data sets (WADSs), and system log data sets (SLDSs) that are required as input to restart can be dynamically allocated.

All data sets using dynamic allocation must be cataloged, except an IMS Monitor data set, which must not be cataloged. A data set that is initially allocated with JCL can be dynamically deallocated and reallocated during the execution of the control region.

Extended Terminal Option

The Extended Terminal Option (ETO) is a function of IMS TM that can be included at system definition. With ETO:

- You can add VTAM terminals to IMS without redefining the system.
- You can dynamically add user LTERMs and remote LTERMs (for MSC links) to IMS.
- You cannot define the master terminal using ETO.
- You cannot use the XRF surveillance link with ETO.
- You must install ETO as a function of IMS and must specify it as part of the system definition in the IMSCTRL macro.

Related Reading:

- For more information on specifying ETO at system definition, see “Chapter 3. Defining Your System” on page 43.
- For information on ETO system security features, see “Chapter 4. Establishing IMS Security” on page 115.

APPC

IMS supports APPC conversations in two scenarios:

- APPC / IMS
- The explicit CPI-C driven interface

The two scenarios differ in the subsystem that manages the updating and synchronization of protected resources that the application program accesses.

In the APPC / IMS scenario, when SYNCLVL = NONE or SYNCLVL = CONFIRM, IMS is the synchronization-point manager. When SYNCLVL = SYNCPT, RRS / MVS is the synchronization-point manager.

In the CPI-C driven scenario:

- The sync point manager is the Resource Recovery Service / MVS (RRS / MVS) function of OS/390.
- The resource manager is IMS.
- The program that accesses and updates the protected resources is the APPC / MVS application program.

Advanced Program-to-Program Communication/IMS (APPC / IMS) defines the formats and protocols for program-to-program communication. APPC / IMS enables applications to be distributed throughout the network and to communicate with each other regardless of the underlying hardware architectures and software environments. APPC / IMS provides a facility for implementing logical unit type 6.2 (LU 6.2) support.

APPC /MVS is used for all interaction with remote LU 6.2 devices or subsystems. IMS accesses the session through APPC / MVS services. Using APPC / IMS, IMS and LU 6.2 devices access each other without requiring coding changes to existing application programs. With slight modifications to IMS application programs,

Common Programming Interface (CPI) communications-driven application programs can communicate with IMS application programs (and can execute as IMS application programs).

A restriction exists, however, on LU 6.2 synchronous conversations with implicit transactions. If a transaction spawns more than one daughter transaction, which, in turn, might spawn other transactions, and one of the daughter transactions provides the response, then the result is unpredictable. In some cases, depending on the execution sequence of the transactions, the LU receives a DFS2082 message and the response is sent to the default TP name DFSASYNC. In other cases, the LU receives the response and no DFS2082 message is issued.

Related Reading: For more information on APPC / IMS, see *IMS/ESA Administration Guide: Transaction Manager*.

Security for Dependent Region Processing

Although security checking can be carried out by terminal and transaction authorization, you can also specify the resources that a region can access by using IMS resource access security. A region is authorized to process a group of transactions identified by an application group name (AGN). The access profile can include the PSBs, transactions, and input LTERMs that are valid for the AGN. The valid resources are declared in the input for the IMS Security Maintenance utility (SMU). You can also use the Resource Access Control Facility (RACF) or an exit routine to authorize the operation of the region.

RACF and SMU provide two different ways to configure your IMS network security profiles:

- Using RACF, you can design security profiles based on user ID.
- Using SMU, you can design security profiles based on LTERM names.

MPP Scheduling

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

When an MPP region has been initialized, it can execute an application program within its virtual storage. By using the scheduling algorithm, the control program selects a message for processing. Using the transaction code, the appropriate application program is loaded into dependent region storage from the IMS.PGMLIB data set. The application program is identified by the PSB name that was declared in system definition to be associated with that transaction code. The convention used by IMS TM for MPPs is that the application program name is the same as the PSB name. The first message segment is then made available from the message queue, and control is passed to the application program.

The scheduling algorithm also controls the amount of processing performed by the application program. You can specify a limit to the number of messages processed in the scheduling of one program. When this number is reached, IMS TM does the following:

- If any equal or higher priority transactions are queued, IMS TM terminates the application program. The region becomes available for another program to be scheduled into its storage. IMS uses the scheduling algorithm to choose the program to schedule.

- If no equal or higher priority transactions are queued and messages are still queued for the current application program, the region goes through quick reschedule and returns the next message to the application program.
- If no more messages exist for the scheduled transaction, IMS TM determines if other work for the region is ready to be processed.
- If no additional work is ready to be processed, IMS TM determines if the region can become pseudo wait-for-input (pseudo WFI). This determination causes one of the following actions:
  - If the region is eligible for pseudo WFI, the region remains scheduled for the transaction and waits until another message is entered for the region. If the next message is for the scheduled transaction, the message is passed to the application program. If the next message is for a different transaction, IMS TM terminates the application program and schedules a new application program to process the new message.
  - If the region is not eligible to become a pseudo WFI, IMS TM tells the application program that no more messages exist, and the application program terminates.

The master terminal operator directly schedules batch message programs by the entry of the JCL to start the batch message region. The program and PSB to be used are explicitly given in the EXEC statement.

Transaction Scheduling

IMS TM schedules a program even when some of the full-function databases that the program can access are not available. When dealing with unavailable data, the program can be sensitive or insensitive to data unavailability.

To be sensitive to unavailable data, the application program must issue the INIT call to inform IMS that IMS should return a status code in the PCB if the call requires access to data that is not available. The program can then take the appropriate action.

If the program has not issued the INIT call, the program is insensitive to data unavailability. When a program requires access to data that is not available, IMS terminates the program with a U3303 pseudoabend and backs out any updates the program has made. The input message that the program was processing is placed on the suspend queue. A separate suspend queue exists for each transaction type. If IMS determines that most messages are failing and being placed on the suspend queue, IMS stops processing that transaction type. When the transaction is started, or when a database used in processing the transaction is started, the messages on the appropriate suspend queues are transferred to the normal queue, and another attempt is made to process the message.

Related Reading: For more information on the suspend queue, see “Scheduling Transactions Using the Suspend Queue” on page 68.

Fast Path

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

Use Fast Path to improve performance for simple transactions. When data communication requirements are for a high transaction volume with rapid database updates and inquiries, the Fast Path facilities offer several advantages over full-function DL/I processing. Examples of application programs with these requirements are the teller transactions in banking and point-of-sale transactions (inventory update) in retail marketing. Fast Path input and output messages use expedited message handling, bypass message queuing, and priority scheduling. Most terminals have Fast Path execution potential. However, terminals that cannot run in response mode do not have Fast Path potential.

For a DB/DC environment, Fast Path requires the Database Manager and Transaction Manager and becomes an integral part of the IMS online system. The control program manages concurrent processing of Fast Path and DL/I programs.

Related Reading:

- For more information on the design, definition, initialization, monitoring, or tuning of databases used with Fast Path, see *IMS/ESA Administration Guide: Database Manager*.
- For information on Fast Path application programming, see *IMS/ESA Application Programming: Database Manager*.

Fast Path Databases

In addition to DL/I databases, two other database types are available with Fast Path: the main storage database (MSDB) and the data entry database (DEDB). These databases are designed for the kinds of application programs that require high availability. The two types offer a choice of either rapid response within high activity or partitioned access within a large volume of data.

Related Reading: For a detailed description of the design advantages and implementation of these Fast Path databases, see *IMS/ESA Administration Guide: Database Manager*.

Dependent Region Use for Fast Path

The majority of Fast Path processing programs are similar in function to message processing programs (MPPs). Message-driven programs correspond to MPPs, and execute in a Fast Path dependent region (IFP region). These programs execute in wait-for-input mode, so that the program execution is equivalent to a dependent region operation. Parallel scheduling is supported, so that another copy of the program can execute in another dependent region.

Because of the ability to perform much of the data entry database maintenance online, such as reorganization and recovery-related functions, your IMS online system should allow for the scheduling of a Fast Path utility region.

Fast Path application programs and utilities can be active concurrently with message processing programs or BMPs. An IMS online system that is using Multiple Systems Coupling (MSC) can also be processing Fast Path transactions. However, message input received through MSC links cannot be directed to a Fast Path application program, nor can they be passed to the Fast Path input exit routine. This restriction does not apply to message input received using Intersystem Communication (ISC) connections.

Fast Path Transactions

A Fast Path application program is driven by transactions that bypass IMS input message queue handling. A transaction can be declared to be Fast Path exclusive. After initial edit, the input message is passed to an exit routine. This routine helps determine the dependent region in which the transaction is executed. The message

is added to a Fast Path message-handling area in the program's storage, and then the transaction is made available to the message-driven program without I/O to the message queues.

A transaction can also be declared as having Fast Path potential. After entry, the transaction is also passed to the user exit routine, which decides whether the transaction should pass directly to the message-holding area in the control program's storage, or whether it should be routed to IMS for normal message queue handling. The queue bypass again leads to the transaction being presented to a message-driven program.

The control of Fast Path messages within the control program's storage is called expedited message handling (EMH). One of the checks it performs is to ensure that messages meet the restriction that they use single-segment input and output messages. The Expedited Message Handler Input Routing exit routine is DBFHAGU0. IMS can use EMH buffers for complete input editing of both Fast Path and full-function transactions.

Related Reading: For more information on DBFHAGU0, see the following publications:

- *IMS/ESA Administration Guide: Transaction Manager*
- *IMS/ESA Customization Guide*

DBCTL Considerations

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

In a DBCTL environment, Fast Path provides improved performance and data availability to programs that can use data entry databases (DEDBs). A DBCTL environment supports only the Fast Path facilities that are related to DEDBs. You cannot run main storage database (MSDB) facilities.

Dependent Region Use for Fast Path: BMPs or CCTL threads can schedule a PSB to access DEDBs. Parallel scheduling is supported; another copy of the PSB can execute in another BMP or CCTL thread. Fast Path application programs and utilities can be active concurrently with BMPs.

Because much of the DEDB maintenance (such as reorganization and recovery-related functions) can be performed online, your IMS DBCTL environment should allow for the scheduling of a Fast Path utility region.

Automated Operator Application Programs

An application program that can issue a subset of IMS operator commands using the DL/I CMD or ICMD calls is called an automated operator (AO) application program.

The CMD call can be used in the following environments:

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The ICMD call can be used in the following environments:

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	X

When the CMD or ICMD call is issued, the operator command is executed and the first segment of the command response is put in the AO application program's I/O area. Subsequent parts of the response are obtained by GCMD (if CMD is issued) or RCMD (if ICMD is issued) calls.

To maintain security, you need to decide which AO application programs can issue operator commands and which commands they can issue. An AO application program can issue a single command or a series of commands.

Related Reading: For information on securing the CMD and ICMD calls, see "Chapter 4. Establishing IMS Security" on page 115.

System Logging and Processing Continuity

To protect the integrity of the data, the online IMS system uses both external security checking and various internal techniques to record the transactions entered into the system and the database update activity. The principal tool for recording online system activity is IMS system logging. Data stored on various logging data sets contains information used for restart, recovery, statistics, and audit purposes.

IMS log data is recorded in four kinds of data sets:

- Online log data set (OLDS)
- Write ahead data set (WADS)
- Restart data set (RDS)
- System log data set (SLDS)

The online system uses a minimum of three OLDSs, one WADS, and a single RDS, all residing only on DASD. When one or more online log data sets are filled, you can archive them to system log data sets using the IMS Log Archive utility. DASD or tape media can be used for SLDSs. Batch systems use system log data sets and are able to log to either tape or DASD.

The online system uses the OLDSs in wrap-around fashion. If dual logging of the OLDSs is an installation requirement, a pair of data sets (primary and secondary) must be assigned. The ddnames for the OLDSs begin with the character string DFSOLP for the primary data set and DFSOLS for the secondary data set. A unique suffix (00 through 99), called an "OLDS identifier," completes the 8-character ddname. Either single logging or dual logging is performed, as determined by DD statements during system initialization or by instructions included in the DFSVSMxx IMS.PROCLIB member.

A WADS is a small data set containing a copy of log records that are in OLDS buffers but have not yet been written to the OLDSs. When logging to DASD (required for online processing), fixed-length blocks make direct retrieval easier. A WADS allows large fixed-length blocks (in variable blocked format) to be written to the OLDSs without the requirement to rewrite blocks. When log data has been written to the OLDSs, the WADS is reused.

If a system failure occurs, the log data in the WADS is used to close the OLDSs. The close process occurs as part of an emergency restart or as an option of the Log Recovery utility.

Write-ahead support is provided for a spare WADS. When a write error is detected, a spare WADS replaces the WADS that encountered the error. Dual WADS logging is also supported if it is required to have backup in the event of a read error while closing the OLDSs from the WADS.

The online IMS system controls the log data sets that are used for startup. It makes use of entries in the checkpoint identification table written on the restart data set, and of log data set information recorded in the DBRC RECON data set. If you are using automatic restart, the `/START IMS` command issued from the system console causes the appropriate kind of restart. Normally, this r

Checkpointing

Checkpointing is the primary technique that IMS uses to record information that can be used to restart an interrupted operation. Using the status information recorded during a checkpoint, IMS restores the contents of the message queues and database changes. Checkpoints are an integral part of system shutdown and startup. Also, the amount of reprocessing, back from the point of system interruption and forward to a continuation point, is reduced when checkpointing is reasonably frequent. Some processing overhead is associated with checkpoint information, but this is an acceptable trade-off for the efficient restart of the system.

In an XRF complex, SNAPQ checkpoint records taken on the active IMS subsystem are used to build control blocks on the alternate IMS subsystem during the synchronization phase.

IMS internal checkpoints are scheduled to occur automatically at predetermined intervals. The interval is specified in terms of an increment to the number of system log records created. As the online IMS events are logged with individual log record types, a count is maintained. When the increment exceeds the specified value, checkpoint processing is invoked. IMS checkpoints can also be invoked explicitly by the master terminal operator and by application programs that have been authorized to issue the `/CHECKPOINT` command.

Fast DB Recovery regions monitor checkpoint records on IMS subsystems, and uses them during database resource recovery.

Locking Mechanisms and Database Integrity

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

IMS offers a choice of locking; you can use program isolation (PI) locking or the services of the Internal Resource Lock Manager (IRLM). The IRLM component is used as an integral part of data sharing, as described in “Chapter 10. Administering a Data Sharing Environment” on page 305. With program isolation, all activity (modifying the database and creating messages) of an application program that is active in the DB/DC environment is isolated from any other application programs that are active in the system. The isolation persists until that application program confirms, by reaching a synchronization point, that the data it has modified or created is valid.

The locking mechanisms are also used to:

- Remove the effects of an abnormally terminated application program
- Perform the processing required for a ROLL, ROLB, or ROLS call
- Resolve deadlock situations

For all the above processing, the removal of database updates and held output messages is done from the previous synchronization point up to the current status. A *synchronization point* is defined as the point at which an application program can be restarted. The first such point for an application program is its initial scheduling. The most common synchronization point is when a GU to the message queue occurs. By issuing a call for the next message, a program in single message mode is indicating the start of a cycle of processing and the completion of any previous work. At this time, any output messages that are queued to a temporary destination are sent to their final destination, and database updates are committed.

An application program can also issue a CHKP call, which forces a synchronization point. For application programs executing in multiple message mode, or BMPs that are not transaction driven, the synchronization point is the time of either the initial scheduling or the last CHKP call.

Another aspect of program isolation is the control of database updates at the segment occurrence level. During the scheduling process, IMS analyzes the intent of an application program toward the database it uses. If a conflict exists with the database usage of a currently scheduled transaction and a candidate for scheduling because an application program needs exclusive use of the database, the scheduling process must select another transaction code and try again. If exclusive intent is not a factor (this is usually the case), application programs are scheduled concurrently. IMS controls the interleaved ownership of database segments with a locking mechanism. As application programs execute, they enqueue on the database records and release those resources, either after update or when the application program reaches a synchronization point.

Possible deadlock situations are resolved in a manner transparent to application programs and terminal operators. When IMS detects a deadlock situation, one of the application programs involved in the deadlock is abnormally terminated with a special abnormal termination code. The abnormal termination causes the activity of the terminated program to be dynamically backed out to a previous synchronization point. Its held resources are released. This allows other application programs to complete their processing. The special code causes the transaction that was being processed to be saved. The application program is rescheduled.

In DBCTL, if a deadlock situation forces an abnormal termination of a CCTL thread, that thread is not saved or retried by DBCTL. The CCTL, upon receiving certain deadlock termination codes, retries its transaction.

If a BMP is selected to be dynamically backed out, it cannot be rescheduled and terminates at its latest synchronization point. If the BMP did not access the message queues for input or issue CHKP calls, the BMP terminates during scheduling and all the BMP database update activity is nullified.

If an ODBA thread is active when IMS DB shuts down or terminates abnormally, the ODBA application thread is terminated. The ODBA application program is not terminated, but it can no longer make calls on the ODBA thread.

Using Data Capture Exit Routines

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	
Note: The Data Capture exit routine is not available to CICS. DBCTL can use the exit routine, but only for BMPs.			

If your installation contains both IMS DB and DB2 databases, duplicating data in IMS DL/I and DB2 relational databases might be required. For example, your DB2 application programs, written in Structured Query Language (SQL), might require data from the IMS DB database. You might be converting your site to DB2 on a gradual basis, or you might want to take advantage of DB2's relational technology for some of your IMS data.

To duplicate data between the two types of databases, you must ensure that each update to data segments occurs in both databases in a timely manner. The process of duplicating updates from an IMS DB database to a DB2 database is known as data propagation. The two ways to propagate data from IMS DB to DB2 are:

- DataPropagator Nonrelational (DPropNR). DPropNR is an IBM licensed program that provides support for data propagation and for exit routines. If you use DPropNR, refer to the product documentation for details:
  - *DataPropagator NonRelational MVS/ESA An Introduction*
  - *DataPropagator NonRelational MVS/ESA Administration Guide*
  - *DataPropagator NonRelational MVS/ESA Reference*

  The description of data propagation in this book is limited to the systems requirements and considerations for the IMS Data Capture exit routine. For information about system requirements, see "Satisfying System Requirements for Data Propagation" on page 111.
- Data Capture exit routine. This is an exit routine that you write to establish a routine for data propagation. It can be written in assembler language, C language, COBOL, or PL/I, and it is called by an application program that requires data propagation.

Related Reading:

- For information on the database considerations associated with the Data Capture exit routine, see *IMS/ESA Application Programming: Database Manager*.
- For information on writing a Data Capture exit routine, see *IMS/ESA Customization Guide*.

The MVS Automatic Restart Manager (ARM)

The MVS ARM restarts a subsystem (or job) after an MVS hardware or software failure.

You can also use ARM to define restart groups. In addition, in the event of an MVS hardware or software failure that requires you to move a subsystem from one MVS system to another, ARM will move all the subsystems defined in the same restart group **as a group** to a remaining MVS system.

IMS supports ARM in these environments:

- TM-DB
- DCCTL
- DBCTL

- XRF
- FDBR

DL/I, DBB, and IMS utilities are **not** supported. The IMS control region will be the only region restarted by ARM.

Attention: The DL/I SAS and DBRC regions are started internally by the IMS control region. IMS dependent regions are not automatically restarted, because they are normally restarted after the IMS control region has restarted.

The element name that IMS uses on the registration call to ARM is the IMSID. The element type is SYSIMS. Duplicate element names are not allowed by ARM. When ARM is used, the IMSIDs of online systems and FDBR systems must be unique.

ARM provides a default ARM level of 1 for SYSIMS.

The IMSID must be unique across the sysplex. ARM will try to move IMS to a surviving MVS if a failure occurs on the MVS or the CPC on which the IMS is executing. If the IMSID is not unique, ARM might move the IMS from the failing CPC to one that already has an IMS with the same IMSID.

If IMS is canceled by MVS, IMS is only automatically restarted by ARM if the ARMRESTART option is specified on the CANCEL or FORCE command.

IMS maintains the following user abend table and de-registers from ARM any time one of these abends occurs:

1. U0020: USER 20 - MODIFY
2. U0028: USER 28 - /CHE ABDUMP
3. U0604: USER 604 - /SWITCH
4. U0758: USER 758 - QUEUES FULL
5. U0759: USER 759 - QUEUE I/O ERROR
6. U2476: USER 2476 - CICS TAKEOVER

The first three of these abends are the result of operator intervention. The last three abends require some external changes before IMS can be restarted.

System Administrator Concepts

Chapter 2. Documenting Your IMS System

When planning to support the administration of an online IMS system, you must consider several responsibilities that involve documentation. Reviewing this documentation for application requirements is a necessary task when you are designing the IMS online system or responding to required changes in that design.

In this Chapter:

Extracting Requirements for Your IMS System

Analysis of the scope and impact of an application in the online environment occurs during the design phase, as well as when application changes are proposed. You must assess the detail of the application requirements by reviewing the following sources:

- Program specifications and logic
- Implementation plan
- Summary of business requirements
- Design change requests

When you examine application documents, you must extract several kinds of information:

- Requirements for IMS function
- Database requirements
- Predictions of the application workload
- Network definition requirements
- Security considerations
- Operating requirements
- Audit and history recommendations
- Performance factors
- Terminal requirements
- FPBUF requirements

If your system must use intelligent remote stations, you must:

- Select features of IMS that can be used to support distributed application processing (for example, ISC, MSC, LU 6.2).
- Identify the terminal support provided by IMS (including ETO).
- Identify the Fast Path requirements of your system.
- Evaluate the off-loading of application requirements to intelligent remote stations and the use of special components or screen formatting.
- Assist in the design of programs that reside in intelligent remote stations and communicate with IMS; identify the Systems Network Architecture (SNA) protocol to be used by IMS and the intelligent remote station.

Participating in Design Reviews

As the development of an application package progresses, reviews of the design should be held.

As the administrator of the online IMS system, you are concerned with adequate detail in the specifications. You need to plan for, and subsequently specify, the online system. Table 2 summarizes the kind of information that you must gather and how that information relates to system administration tasks.

In addition to application design specifications, the application development team might maintain a controlling document for schedules and responsibilities. You must contribute to this plan with your own requirements and milestones, such as completion dates and testing dates for the operating procedures.

Table 2. Administration's Use of Design Reviews

Design Stage	Information Needed	Administrative Tasks
Design review 1	Scope of project Hardware/software requirements End-user/development contacts	Analyze IMS function requirements Contribute to documentation plans Check standards compliance Assess network impact Assess DP operations impact Make workload predictions
Design review 2	Use of MFS and screen usage characteristics Elements of end-user control Network planning Pointers for operator control Transaction workload	Establish MFS library control, identify format names Coordinate dictionary use Check naming standards Track network requirements Plan security strategy Begin RTO and MTO procedures Predict processing workload
Design review 3	Message definition Conversational attributes Databases and programs System resource requirements Need for message edit Recovery considerations Security and audit	Calculate message queues Calculate SPA data Specify system and JCL Finalize system requirements and hardware plan Specify message edit coding Begin recovery procedures Develop security design
Database Application design review	Database maintenance Database sharing Validation/acceptance plans Performance predictions Monitor plans	Document online databases and image copy requirements Choose system integrity options Establish system availability Establish performance criteria and plan monitoring
Logic review	Monitor pointers Program preload Virtual storage needs Database processing intent conflicts	Develop a monitoring strategy Plan dependent regions Develop scheduling algorithm Estimate buffer pool and system data set resources

Related Reading: For more information on the purpose and scope of design reviews, see *IMS/ESA Administration Guide: Database Manager*.

Establishing Naming Conventions

A critical part of the application specification and the control of the IMS online system design is maintaining naming conventions for your resources. When you define a large system that has many resources, the ability to recognize the characteristics of the resource by its name has many advantages:

- The system definition input is easier to check, and the identification of changes is easier.
- The MTO control is more effective and efficient and less prone to error.
- The modification of the application design can more easily recognize already defined resources rather than creating ambiguous or unnecessary additional resources.

You should establish naming conventions, in cooperation with database administrators, for at least the following resources:

- Databases, their ddnames and data set names
- Image copy and change accumulation data set names
- Segment and field names
- PSB and program names
- Transaction codes
- MFS format names
- LTERM and node names
- ETO terminal and user names
- LU 6.2 descriptor names
- Online log data set names
- System log data set names
- IMS Monitor output data set names
- Link names and IMS system IDs (for Multiple Systems Coupling)
- Fast DB Recovery region names

Table 3 shows some examples of naming conventions that can be applied to resources controlled by IMS for online applications.

Table 3. Examples of Naming Conventions

Resource	Naming Convention	Description	
Transaction	Taaatsss	**T**	Transaction
		aaa	Application identifier
		t	U for update, or R for inquiry transactions
		sss	Transaction sequence

Table 3. Examples of Naming Conventions (continued)

Resource	Naming Convention	Description	
LTERM name	cnnxiiii	**c**	L for local, S for switched, N for non-switched
		nn	A 2-character code for terminal type
		x	A 1-character attribute indicating the screen size, printer, or component
		iiii	A 4-character identifier
MFS (MSG name) (MID and MOD)	aaaiiii	**aaa**	Application identifier
		iiii	A 4-character identifier
MFS (FMT name) (DIF and DOF)	aaaiii	**aaa**	Application identifier
		iii	A 3-character format identifier
Module name	Maaaiiii	**M**	Module name
		aaa	Application identifier
		iiii	A 4-character identifier
Job name	Jaaannnn	**J**	Job name
		aaa	Application identifier
		nnnn	A 4-character job identifier

More information on naming conventions is available as follows:

- *IMS/ESA Administration Guide: Database Manager* for recommendations for naming conventions for databases, PSBs, and programs.
- *IMS/ESA Installation Volume 2: System Definition and Tailoring* for a list of restricted names.
- *IMS/ESA Administration Guide: Transaction Manager* for specific naming conventions for ETO.

Using a Data Dictionary

If you use the IBM MVS DB/DC Data Dictionary licensed program (program number 5740-XXF), you can use several of its features to assist in system and network documentation. The product supports interactive processing, enabling you to use a terminal to create the documentation at your convenience.

The product provides standard categories for databases, segments, fields, PSBs, PCBs, transactions, programs, and the system. You can use these categories to build up detailed descriptions of the online IMS system resources. Procedures and execution JCL can also be recorded (as User Data). Three of the stage 1 macros—APPLCTN, DATABASE, and TRANSACT—can be produced as card output.

You can use the Extensibility Feature to define terminal subjects and specify a selection of attributes to record for each terminal. With the Description and User Data segments, other free-form text can be associated with the terminal subjects.

Documenting Your System Characteristics

You must create an independent body of information that documents the design and operation of the production system. This material is derived from the application requirements and includes the intended use of IMS facilities. The remainder of this chapter describes documentation activities for the following:

- IMS system definition
- IMS network
- Terminal profiles within the network
- Transaction profile names
- Configuration of the production system

IMS System Definition

The detail of your design for an IMS online system is reflected in the system definition macro specifications. The input to the first stage of system definition processing can be used as a major documentation tool. Including comments on the choice of parameters should help you control definition changes.

Related Reading: For more information on this aspect of system documentation, see “Managing System Definition” on page 45.

The IMSCTRL macro statement offers an ETO option that creates a report of current ETO descriptors in the network. You can use the ETO descriptor report to monitor your current network definition.

Related Reading: For more information on the IMSCTRL macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

IMS Network

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Within the broader scope of system documentation, you need to document the details of the IMS network. Start this documentation as soon as initial plans for an application system become available. The advantage to starting early is that you can become familiar with the physical network and the way it will logically be used by IMS connections. When your online IMS system uses terminals that are part of a network defined to VTAM, you might be able to use some of the documentation developed by the system programming staff.

Documenting in detail allows you to:

- Familiarize yourself with the features and operation of the terminal devices
- Find out if the application is going to use the terminal in an unusual way and, if so, research the potential problems
- Prepare for IMS system definition stage 1 input
- Prepare ETO descriptors
- Prepare LU 6.2 descriptors

- Plan for the installation and network generations
- Understand the operational aspects of subsets of the total configuration

Terminal Profiles

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

One way to document the required network, after the major design is stable, is to record the intended use and characteristics of each terminal. You construct a profile that contains:

- The terminal type, its required features, and the type of connection it is to use
- What options were chosen for the device and the reason for the choice
- The characteristics of how the terminal is to be used by the application
- The proposed LTERM names that can be associated with the terminal or the user
- If a VTAM-supported device, the node name and transmission characteristics
- The extent of the proposed usage and whether the device is to be shared with non-IMS users
- The diagnostic procedures that are appropriate for the device
- User profiles and user profile security

For dynamic terminals, you should maintain a record of the characteristics of dynamic terminals and users to make future changes easier.

Transaction Profile Names for APPC/IMS

Definitions of transaction program names (TPNs) are contained in the APPC/MVS resource, TP_Profile. Within TP_Profile, TP profile data sets provide attribute information for each TPN. You can define TPNs that have different characteristics for each LU name with which they are associated.

Related Reading: For more information on TP names, see *IMS/ESA Administration Guide: Transaction Manager*.

Configuration of Production System

You should create a configuration map showing how individual terminals and clusters of terminals are to be connected in the network. This gives you some insight as to how to specify the network control to the MTO. Having the configuration map is also an aid when you interact with various technical specialists. The map should show:

- Processors or host computers
- Channels and lines (including the type of line)
- Communication controllers
- Control units and the terminal attachments
- VTAM node names
- XRF USERVAR
- Alternative connections or configurations

To solve problems that might arise, you must be able to identify the terminal or control unit that has the problem. You can assist both end users and service

personnel by placing identifying labels on the devices. In addition to the IMS address or node name and the LTERMs appropriate for that terminal, include the hardware address, circuit identification, and other data that might be necessary.

Chapter 3. Defining Your System

This chapter describes the system definition macros and their required and optional control parameters.

In this Chapter:

When you design an IMS online system to meet the requirements of application programs, you must specify IMS control information. Do this by:

- Using system definition macro statements to specify IMS online functions and system control options
- Tailoring a set of execution JCL
- Tailoring virtual storage

The total of these specifications represents the overall design and control of the IMS online system. This chapter describes the system definition macros and their required and optional control parameters, in the context of:

- Managing the system definition process
- Declaring what application programs and transactions can operate
- Describing the physical communication network
- Deciding which control and integrity factors to use
- Initializing IMS system data sets
- Tailoring initial JCL procedures
- Specifying IMS execution parameters

Each decision contributes to the overall design. Many of the parameters are dependent on the IMS functions required by the application programs, and more particularly on the projected workload. The decisions you make are made in conjunction with design decisions made for databases and applications.

How System Definition Is Related to Installation

System definition and JCL preparation are only a part of the total installation planning process.

Related Reading: For step-by-step instructions on installing an IMS system, see *IMS/ESA Installation Volume 1: Installation and Verification*.

A full installation process includes:

1. Building IMS system libraries
2. Allocating and cataloging IMS data sets
3. Defining the IMS system
4. Preparing the operating system for IMS, including:
   - VTAM
   - RACF

- APPC/MVS

5. Installing IMS exit routines
6. Tailoring IMS buffers and certain performance options
7. Defining terminals (VTAM and non-VTAM)
8. Updating MFS device characteristics table and MFS default formats
9. Defining LU 6.2 descriptors
10. Defining ETO descriptors
11. Building IMS.DBDLIB
12. Building IMS.PSBLIB
13. Building IMS.ACBLIB
14. Preparing for dynamic allocation of databases and related system data sets
15. Compiling message format descriptions
16. Loading application programs
17. Initial loading of databases
18. Establishing IMS security
19. Initializing IMS.MODSTAT
20. Copying staging libraries to active libraries

This chapter provides information on the following topics from the previous list:

- 2. Allocating and cataloging IMS data sets
- 3. Defining the IMS system
- 5. Installing IMS exit routines

Related Reading:

- For information about building database and PSB libraries, and for dynamic allocation preparation (items 11 through 14), see *IMS/ESA Utilities Reference: System*.
- For the design and definition of message formats, together with details of the supporting utilities (item 15), see *IMS/ESA Messages and Codes*.
- For guidance on loading databases (item 17), see *IMS/ESA Administration Guide: Database Manager*.
- For information on defining LU 6.2 descriptors and ETO descriptors (items 9 and 10), see *IMS/ESA Administration Guide: Transaction Manager*.
- For the design of IMS security (item 18), see “Chapter 4. Establishing IMS Security” on page 115.
- For the remaining items and the detailed specification of system definition as they pertain to preparing the MVS system and initial library build, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For additional specifications for Multiple Systems Coupling, see *IMS/ESA Administration Guide: Transaction Manager*.
- For information on IMS XRF systems, see “Chapter 5. Designing a System with Extended Recovery Facility” on page 159.
- For information on macros or system data set references that apply to these facilities, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For information on planning for the design, installation, and system definition requirements for IMS systems that are to share access to databases, see “Chapter 10. Administering a Data Sharing Environment” on page 305.

Managing System Definition

Whether you are preparing an initial definition of an IMS online system, integrating additional application programs, or making minor alterations to an existing design, the major control point is the system definition source material. The application requirements and your system control options should be organized in a logical sequence that parallels the structure of the system definition structure.

System definition is a two-stage process. Stage 1 checks your input specifications and generates a series of MVS job steps for stage 2. Stage 2 builds IMS system libraries, execution procedures, and the IMS online control program. Figure 8 on page 46 illustrates the two stages of the system definition process.

In an XRF environment, system definition of the active and alternate IMS subsystems must be identical, although two separate system generations can be performed.

Related Reading: For information on defining an XRF system, see “Chapter 5. Designing a System with Extended Recovery Facility” on page 159.

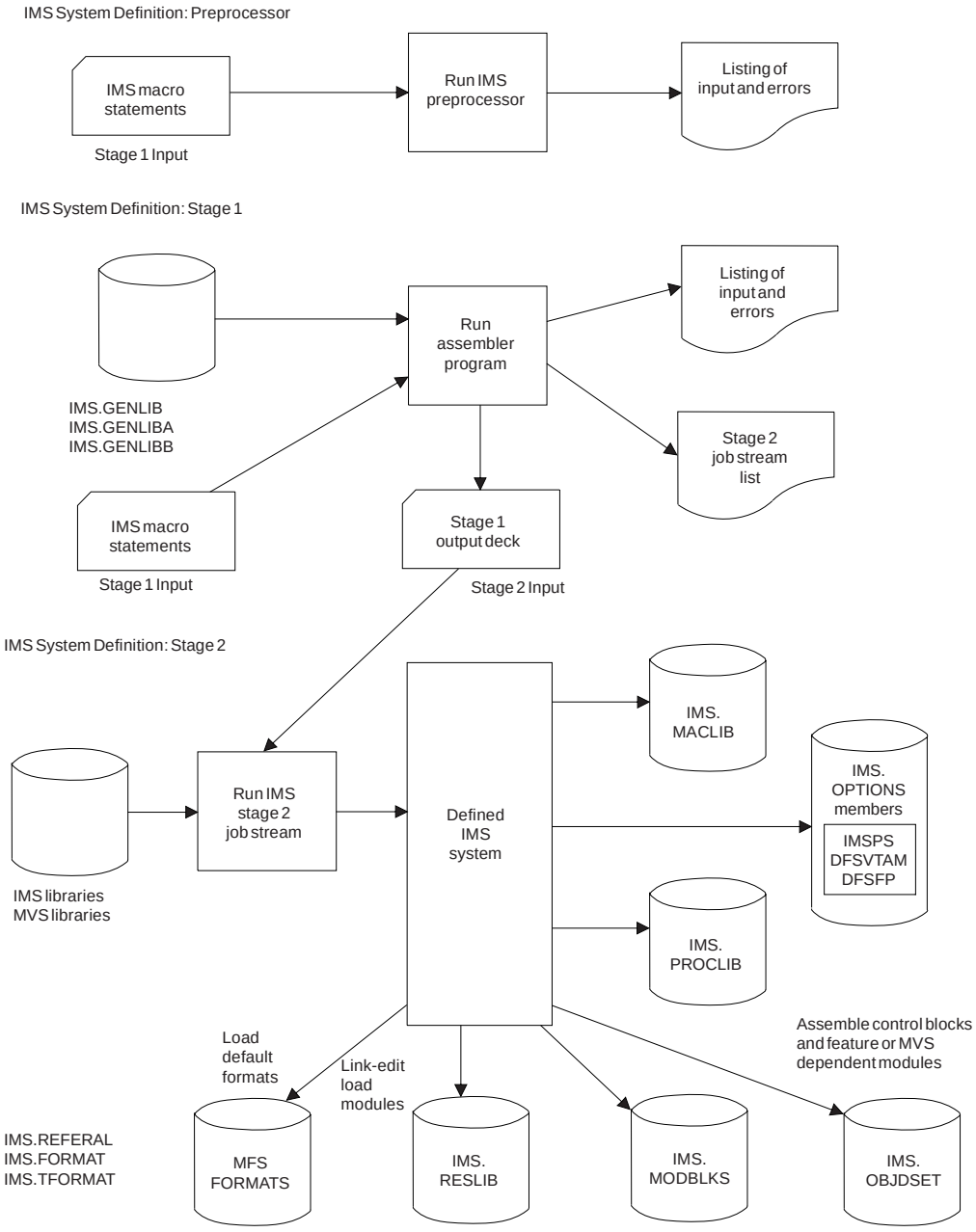


Figure 8. Summary of the Two Stages of System Definition Processing

Structuring Stage 1 Definitions

You use the system definition process to customize the control program to your requirements. The control program is then loaded during the initialization of the IMS online system using a set of macro source statements. Each statement is coded with its own parameters. The composite of all the macro statements is called stage 1 input. times. Some macros are only coded when special support is required, such as that for an IBM 3275 configuration or an attached System/3. Six sets, or groupings, of macro statements make up the content of the stage 1 input. Within each grouping, individual macros specify data to a required function or to a part of the total physical online configuration.

You can view these groupings of macros as a type of hierarchic structure, as shown in Figure 9. The group of required system environment macros is shown as a root segment.

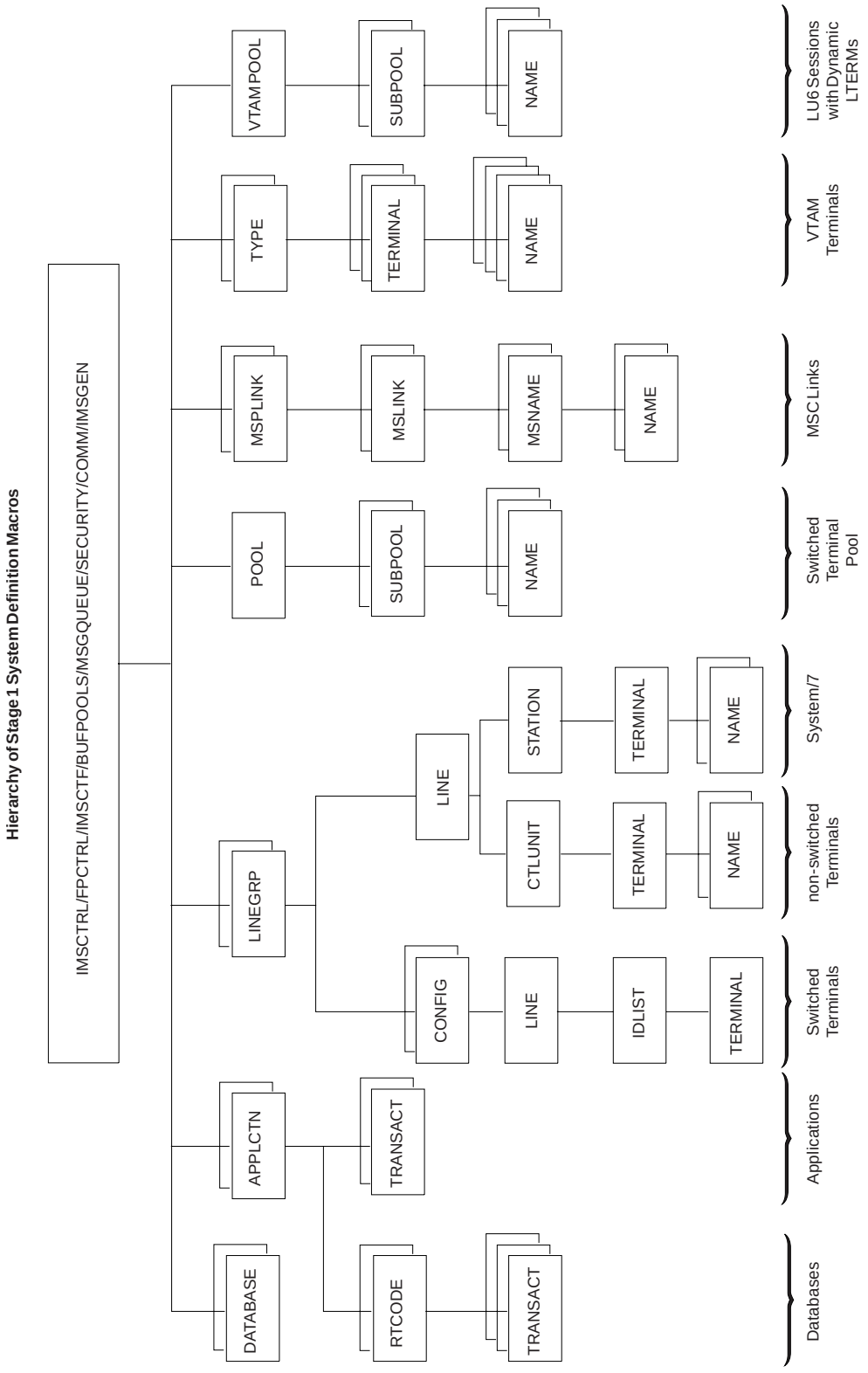


Figure 9. Hierarchy of Stage 1 System Definition Macros

Coordinating System Definition Input Data

The system definition input data you need to coordinate can be organized so that it parallels the sets of macro definitions. Separate macro subsets define the following:

- IMS system environment
- Databases and programs
- Use of non-VTAM terminals and devices
- MSC configuration
- Use of static VTAM terminals

Although the initial installation of an IMS online system establishes relatively stable libraries and MVS operating system options, the system definition portion is subject to change and tuning actions. The stage 1 data set becomes the master control for system definition maintenance. The data set should contain comment statements declaring why certain options are chosen.

Related Reading: For information on the system definition macros and their input data, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Verifying the Stage 1 Input

The stage 1 input contains a structured definition of many resources to be used by the IMS online control program. It is important that you verify the content of the stage 1 input and the accuracy of the macro statement coding. The two kinds of verification are:

- Appropriate uniqueness of resource names
- Consistent and correct macro statement usage

These verifications are described in the following sections.

Resource Name Checking

The names of resources are important not only from a documentation standpoint (conformance to naming conventions), but also from an operational standpoint. An LTERM name could be explicitly used by the MTO or it can be coded in an application program. As such, the resource names must be unique. Also, IMS reserves the use of certain resource names.

Related Reading: For a list of the naming rules and reserved names, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Stage 1 verifies that resource names are valid and appropriately unique. A benefit of this checking is to ensure that conflicts between resource definitions are detected before the control block building that takes place in stage 2. Some installations regularly execute stage 1 to perform this checking.

Performing Stage 1 as a Separate Step

Because definite sequence requirements and dependencies exist between the parameter specifications, the macro statement checking performed in stage 1 is also valuable.

An option is available to help you make efficient use of the stage 1 processing. The NAMECHK parameter on the IMSCTRL macro lets you bypass name checking—assuming you have made sure that no invalid or duplicate resource names exist. If your installation has made use of the optional preprocessor and you are checking a sizeable stage 1 input, specifying NAMECHK=NO can reduce the processing performed by stage 1. You can also specify that the sort not be

performed in stage 1. (Specify S2 as the second value for the NAMECHK parameter.) When you perform the system definition and execute both stage 1 and stage 2, the default of NAMECHK=(YES,S1) is recommended so that full checking is part of the definition process.

You should track the progress of the stage 1 runs and, if necessary, investigate any detected problems.

Using the System Definition Preprocessor

IMS provides a preprocessor that scans the stage 1 input to assist with the necessary name checking. The preprocessor checks for duplicate names among the names you have defined and ensures that they are of the correct length and format. Assigned names are checked across resource types, too, so that transaction codes, LTERMs, and IMS subsystem names (used for multiple systems coupling) do not contain duplications. The preprocessor helps maintain the integrity of the stage 1 input stream.

You can develop exit routines that gain control during the execution of the preprocessor. You can use one or both of the following exit routines:

- Exit DFSPRE60

  This routine gains control after each record in the stage 1 input is read, but before any other processing takes place. It can modify the contents of the record and can even submit further statements to the preprocessor for checking. Any changes made by this routine are not permanent, nor are these changes automatically passed to stage 1.
- Exit DFSPRE70

  This routine gains control when all cross-checking has been completed. It has access to all the tables of resource names. The routine can then format these tables as part of a documentation effort.

Related Reading: For a description of the coding requirements, see *IMS/ESA Customization Guide*.

Planning for Different Types of System Definition

You do not have to perform a complete system definition each time a parameter value changes. Many changes that result from tuning activities can be achieved by using JCL parameters. However, if you add application programs (excluding APPC/IMS), databases, or physical changes (excluding ETO) to the network, redefinition of the IMS system is required. The SYSTEM keyword of the IMSCTRL macro selects different types of system definition.

The next three sections describe activities involved in system definition:

- Choosing a system definition class and type
- Specifying alternative versions of an online system
- Controlling system definition output

Choosing a System Definition Class and Type

The system definition includes a class and a type. The class and type are specified with the IMSCTRL macro and determine the IMS system to be built. The system definition classes are DB/DC, DCCTL, and DBCTL. The DB/DC class builds the standard IMS system. The DBCTL and DCCTL classes build only the subsystem specified.

Related Reading: For more information on the types of system definition specified by the IMSCTRL macro, their typical use, and the resulting effect, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Following your initial system definition, use the ONLINE, CTLBLKS, and NUCLEUS types of generation to implement most of the required changes. These generations require a cold start of the IMS online system for the changes to take effect.

For certain modifications and additions, you can take advantage of the online change method using the MODBLKS generation. The changes are made active during the execution of the online system and do not require a restart operation. Managing this kind of change is described in "Chapter 9. Modifying Your System Design" on page 295.

Specifying Alternative Versions of an Online System

A special capability exists for you to generate different versions of the IMS online system. The control modules in the nucleus and control blocks carry a 1-character suffix. The value specified for the control region parameter SUF= controls which IMS configuration is to be executed. The default value is 0 (zero). You control the generation of this suffix with the SUFFIX keyword on the IMSGEN macro.

After all generations, you can optionally execute the Security Maintenance utility when you are using IMS security in the system. The EXEC statement for the Security Maintenance utility must specify or accept the default to UPDATE and include the 1-character nucleus suffix so that the security matrix tables are merged into the corresponding regenerated nucleus.

Controlling System Definition Output

The SYSTEM keyword on the IMSCTRL macro determines the extent of stage 2 processing. You can use the SYSTEM keyword when you are using ETO.

A successful stage 1 execution generates an output deck that consists of an extensive series of self-contained jobs. Warning messages are also embedded within a listing of your input source. A trailer to this listing contains instructions on the content and execution of the stage 2 job stream. Stage 2 processing builds a control program, or nucleus, which is tailored to your required IMS function. The control program, along with most of the internal control blocks, is placed in the IMS.RESLIB data set.

Because the amount of stage 2 processing time is significant, you should review the stage 1 messages carefully. A misplaced NAME macro or misspelled LTERM name can cause service to be unavailable to an end user and require an additional stage 2 execution.

Large System Generation

The reason for choosing a large system generation is to generate a large number of IMS resources. As new resources are defined, more storage is required to process these definitions during stage 1 processing. Unfortunately, for increasingly large systems, all the private storage in a 9 MB region can be exhausted before the stage 1 assembly can complete, causing the assembler to abend. In a case such as this, you can choose a large system generation (LGEN). LGEN does not support the building of ETO descriptors. If you need to build ETO descriptors, run the LGEN first (Stage 1 and Stage 2). Then, run a standard GEN (CTLBLKS) specifying ETOFEAT=(YES,YES,ONLY).

Online Change

It is possible to make system definition changes without shutting down the system. If you follow the rules for online system change when creating the LGEN system definition, you can use the online change process to incorporate those data sets built from the LGEN system definition described in “LGEN System Definition Changes”. If you do not follow the rules for online change, you must shut down and restart IMS to incorporate the changes.

Related Reading: For more information on online change, see “Chapter 9. Modifying Your System Design” on page 295.

LGEN System Definition Changes

Two steps are required to define a large system generation. First, in the IMSCTRL macro statement, you must specify LGEN as the fourth subparameter of the SYSTEM keyword. You define all other macros for the system generation as you normally do.

Second, you must allocate and catalog two additional data sets. These data sets are required for the stage 1 processing in the LGEN environment. The default names for these data sets are IMS.LGENIN and IMS.LGENOUT.

Related Reading: For more information on these data sets, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

LGEN Stage 1 Processing

The stage 1 processing for LGEN system generation is different than a normal system generation. You must execute the preprocessor to assemble an LGEN generation. A stand-alone assembly of an LGEN generation causes an error.

In turn, the preprocessor performs the necessary resource name checking. The preprocessor uses storage above the 16 MB line, assembling the generation in cycles. This ensures that the application program requires only 4 MB of private storage. You receive two reports from the stage 1 processing: a summary of return codes for each assembly and a summary of error messages.

The result of the assembly cycles is the job stream that creates members of a partitioned data set (PDS). These members are held in the LGENIN data set and are used as input to the Sort/Split utility program. The Sort/Split utility sorts all the PDS members for a given resource type and then splits them into parts that assemble in a 4 MB region. Each part becomes a member of a PDS, stored in the LGENOUT data set. The Sort/Split utility then supplies you with a report about its processing.

The Sort/Split utility processes each resource type individually and includes both standard and customized resources. Customized, or user-defined, resources are normally created by modifying stage 1 macros. A control record is required for each user-defined resource in the Resource Information file member. This record tells the Sort/Split utility to process the resource.

Related Reading: For more information on customized resources and the Sort/Split utility, see *IMS/ESA Customization Guide*.

LGEN Stage 2 Processing

The LGEN type of stage 1 processing creates altered job steps for stage 2 processing. But the effect of this is transparent to the user and to IMS function.

Including Fast Path in DCCTL or DB/DC

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

The primary tasks for system administrators when Fast Path application programs are to be part of the IMS online system are the same as those described in “Chapter 1. Introduction” on page 9. The major additional consideration is the priority that the Fast Path processing is to have over other work. Fast Path is designed to drive relatively simple transaction and database processing through the system at high transaction rates. You must assess whether the combination of Fast Path regions and other regions, as well as the increased requirement for control program storage, can be supported.

The following sections describe activities associated with including Fast Path application programs in your IMS online system.

Gathering Information for Fast Path Execution

When Fast Path application programs are to be part of the IMS online system, you should:

- Gather the application program and Fast Path dependent region requirements.
- Identify the messages and routing codes for message-driven programs.
- Document the MSDB initialization and LTERM ownership details.
- Document the DEDB areas, the data sets associated with each area, and the requirements for online replication and maintenance.
- Obtain the Input Routing exit routine (DBFHAGU0) and document the routing function.
- Prepare the system definition information for the application programs.
- Tailor the online procedures and JCL for Fast Path operation.
- Modify operational procedures for startup, control, and recovery of Fast Path resources.
- Assess the extent of mixed mode and plan a monitoring strategy that includes both Fast Path and DL/I performance data.

Criteria for Fast Path Application Programs

Application programs that use the Expedited Message Handler must meet the following criteria:

- Each transaction is initiated by a single-segment response-type message.
- Each input message requires only one response, a single-segment message, or no response.
- IMS conversational processing is not supported.
- Fast Path programs cannot act as automated operator programs, nor can they issue IMS commands.
- Fast Path transactions cannot be sent over any of the four types of MSC physical links for processing in another IMS system.
- ETO does not support terminal-related or non-terminal-related LTERM key MSDBs.
- LU 6.2 devices cannot access terminal-related MSDBs. LU 6.2 devices can access dynamic MSDBs, but only in read-only mode.

Including Fast Path in the System Definition

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

The Fast Path application programs are described within the body of the system definition input. They require the presence of the following macros:

FPCTRL To include Fast Path facilities

APPLCTN To declare the application programs

RTCODE To direct input messages to Fast Path programs

TRANSACT To declare the transaction code and processing characteristics

You use a system environment macro to specify Fast Path control program facilities in an IMS online system in the same way that you use macros to specify options and buffer sizes.

The FPCTRL macro includes Fast Path facilities in the IMS online system and reserves certain resources for Fast Path. In DCCTL, you need to include the FPCTRL macro in the system definition in order to support Fast Path processing and transactions. However, you do not need to include FPCTRL macro parameters, because these parameters only apply to Fast Path databases.

Related Reading: For more information on the FPCTRL macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

In DB/DC, you can specify the following parameters:

OTHREAD
Keyword specifying how many output threads (1 to 255) are to be used for the asynchronous DEDB update processing. The default is 2.

BFALLOC
Keyword controlling the buffer pool characteristics.

DBFX
Keyword specifying the number of buffers to be set aside for DEDB updates used by sync point processing. These buffers are part of the total number given for DBBF.

DBBF
Keyword specifying the maximum number of buffers available to the online system for MSDB and DEDB processing. The range for both numbers is 1 to 99999. The default values are 4 for MSDBs and 10 for DEDBs.

BSIZ
Keyword specifying the actual size of an individual buffer. The control interval size is 512 bytes, 1KB, 2KB, 4KB, or a multiple of 4 KB (up to 28 KB), depending on the size of the largest CI used for DEDB processing.

Determining EMHB Size for Fast Path

Consider several factors to determine EMHB size:

- EPSESRT size
- EMHB default size
- EMHB allocation

EPSESRT Size Determination: The EPSESRT buffer holds the input message in the IMS Fast Path (IFP) dependent region. One extended program specification table (EPST) exists for each IFP.

The size of the EPSESRT must be equal to the largest of the following:

- The EMHL parameter, if specified. The EMHL parameter can be specified on the EXEC statement of the IMS procedure or on the DFSPBxxx parameter block.
- The largest FPATH value specified on a TRANSACT or APPLCTN macro in the system definition.
- The largest FPBUF value specified on a TERMINAL macro in the system definition.

If none of these is specified, the default size of the EPSESRT is 2KB.

EMHB Default Size Determination at Initialization: The EMHL execution parameter defines the default size of the EMHB buffer. At initialization the default size of the EMHL execution parameter is determined as follows:

- EMHL startup parameter, if specified.
- If the EMHL parameter is not specified, the size of the EPSESRT determines the default size. The determination of the EPSESRT size is explained above in "EPSESRT Size Determination".

EMHB Allocation during Normal Transaction Processing: The first time that IMS schedules a Fast Path transaction for a terminal, it allocates an EMHB buffer for the terminal based on the following factors:

- The EMHB buffer is the larger of:
  - The default EMHB length specified by the EMHL execution parameter.
  - TRANSACT (SMB) specific EMHB length. If an FPATH EMHB size is specified on both the APPLCTN and the TRANSACT macros for the transaction, the transaction-specific size is used.
- A session (except LU 6.2) keeps its EMHB until the session is ended (through logoff or CLSDST, for example). A terminal that has processed one Fast Path transaction is likely to process others.
- During normal processing, if a session needs a larger EMHB for a specific transaction, it increases the size of the EMHB to the larger transaction-specific size. It keeps the larger EMHB until session end.

Special Considerations for EMHB Size: If the input message is too large to fit into the default EMHB length or transaction-specific EMHB length, whichever is larger, the input message is rejected with message DFS444.

If the correct size for the EMHB cannot be obtained for the terminal, the input message is rejected with message DFS3971.

Including Fast Path in DBCTL

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

The following Fast Path information applies to a DBCTL environment:

- Analyzing requirements
- Including Fast Path in the system definition

Related Reading:

- For information on Fast Path EXEC statement parameters for the control region and dependent regions, see “Specifying EXEC Statement Parameters” on page 94.
- For information on tailoring execution procedures for Fast Path, see “Tailoring the IMS Procedure Library” on page 90.

Analyzing Fast Path Requirements

The primary tasks for system administrators when Fast Path application programs are to be part of the IMS DBCTL environment are the same as those described in “Chapter 1. Introduction” on page 9. The major additional consideration is the buffer requests.

Special actions for Fast Path are:

- Gather the application requirements for DEDB use.
- Document the DEDB areas, the data sets associated with each area, and the requirements for online replication and maintenance.
- Prepare the system definition information for the application programs.
- Tailor the online procedures and JCL for Fast Path operation.
- Modify operational procedures for startup, control, and recovery of Fast Path resources.
- Assess the extent of mixed mode and plan a monitoring strategy that includes both Fast Path and DL/I performance data.

Including a Fast Path Environment

The Fast Path application programs are described within the body of the system definition input. They require the presence of the following macros:

FPCTRL To include Fast Path facilities

APPLCTN To declare the application programs

You use a system environment macro to specify Fast Path control program facilities in an IMS DBCTL environment in the same way that you use macros to specify options and buffer sizes.

The FPCTRL macro causes Fast Path facilities to be included in the IMS online system and reserves certain resources for Fast Path. The OTHREAD keyword specifies how many output threads are to be used for the asynchronous DEDB update processing. Specify a number from 1 to 255. The default is 2.

The BFALLOC keyword controls the buffer pool characteristics. Using the DBFX parameter, you specify the number of buffers to be set aside for DEDB updates used by sync point processing. These buffers are part of the total number given for the next parameter, DBBF. This parameter is used to specify the maximum number of buffers available to the online system for DEDB processing. Specify a number from 1 to 99999. The default value is 10. The actual size of an individual buffer is specified by the BSIZ parameter. The control interval size is 512 bytes, 1 KB, 2 KB, 4 KB, or a multiple of 4 KB (up to 28 KB), depending on the size of the largest CI used for DEDB processing.

Extended Terminal Option Descriptors

An installation can use Extended Terminal Option (ETO) descriptors to migrate quickly from static-system definition to dynamic-terminal definition. Logon descriptors contain information relating to the physical characteristics of terminals.

User descriptors are necessary for the creation of user control blocks; they contain information about user options and message queue names. MSC descriptors identify the remote LTERMs that are associated with MSC links. Descriptors can be added, deleted, or updated at initialization.

Related Reading: For more information on ETO descriptors, see *IMS/ESA Administration Guide: Transaction Manager*.

LU 6.2 Descriptors

LU 6.2 descriptors provide information about each LU 6.2 device. Descriptors are built during IMS initialization.

LU 6.2 descriptors provide information required to dynamically create queue control blocks and to set processing options. With user descriptors, you can optionally specify an LTERM that associates an output destination with an LU 6.2 device. You can change application programs using alternate PCBs to use LU 6.2 devices without application program coding changes or application program awareness of the LU 6.2 device type. LU 6.2 descriptor LTERMs are output-only. They are never used by IMS as an LTERM name that is associated with an input message.

Related Reading: For more information on LU 6.2 descriptors, see *IMS/ESA Administration Guide: Transaction Manager*.

Defining Online Applications with System Definition Macros

IMS online applications consist of individual programs scheduled to process transactions when they are entered into the predefined network. (Scheduling must be invoked by JCL for batch message programs.) You must tell the control program which programs are potentially available and the transactions they process. The order in which the transactions are handled is determined by the scheduling function. You must also identify all the databases; this is the cumulative set of databases that could be referred to by the programs. Declarations are made by coding corresponding system definition macros as shown in Table 4.

Table 4. System Definition Macros for Defining Applications

Resource	Identification	Macro	Number Coded
Databases	Physical DBD name	DATABASE	1 or more
Programs	PSB name	APPLCTN	1 per PSB
Transactions	Transaction codes	TRANSACT	1 or more per APPLCTN 0 for non-message BMP

For a DBCTL environment, you define application programs only for BMP regions, using the APPLCTN macro. For a CCTL region, you define the PSB names that the CCTL applications require, using APPLCTN. The TRANSACT macro is not used in defining a DBCTL environment.

For a DCCTL environment, you define application programs for BMP and MPP regions using the APPLCTN macro and the TRANSACT macro. The DATABASE macro is not used in DCCTL.

Declaring Online Databases

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

The IMS control region responds to the database access requirements of application programs scheduled for execution and provides DL/I call services. You need to define a list of all the physical databases that could be used by the programs. You do this by including DATABASE macro statements in the system definition stage 1 input. You need separate statements for the index and data portions of a HIDAM database. You also need statements for any secondary index database that refers to any database defined elsewhere in the set of DATABASE macro statements.

Another attribute of a database is specified with the ACCESS keyword. The default value (EX) specifies that the database is for exclusive use by this online system. If you plan to allow concurrent use by other IMS systems, refer to “Establishing Database Access” on page 307 for an explanation of the other values for the ACCESS keyword.

Declaring Message Processing Programs

The online system identifies an application program by a unique PSB name. The same PSB name is used as the name of the message processing program. The PSB name is coded in the APPLCTN macro. Inherent in the PSB is the access to the message input queue and the declaration of any alternative destinations, other than the input source, for messages sent by the program. The major portion of the PSB defines a complete list of database access intentions (down to the segment level, or to the field within segment). The online system expects this control block to have been predefined. The PSB is prepared for access as a member of IMS.ACBLIB.

Declaring Program Characteristics

You must declare the program to be online or batch message.

Design of the application can result in a program that requires a large amount of virtual storage. If the processing follows a sequence or is staged, the program design can use overlay. You must declare this as one of the PGMTYPE parameters.

You can specify the transaction class for the messages the application program receives, rather than coding it on subsequent TRANSACT macros. Assigning classes is described in “Planning a Scheduling Algorithm” on page 62.

In the APPLCTN macro, you must also specify whether the application program can be scheduled in only one region or concurrently in multiple dependent regions. Remember that scheduling two copies of the same application program to execute in parallel requires that processing be truly independent. Several implications for the use of system resources are:

- Each schedule requires its own section of any shared pools.
- Program isolation activity might increase if the processing has the potential to perform updates in the same database record.
- You cannot control the ultimate processing sequence for transactions of the same type.

Related Reading: For more information on declaring program characteristics, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Choosing PSB Performance Options

The RESIDENT option of the APPLCTN macro allows you to specify whether or not the PSB is resident.

Using the RESIDENT option eliminates I/O to ACBLIB when the PSB is scheduled. No storage fragmentation exists in the resident PSB space as there is in the PSB pool, and the result is a more efficient use of storage.

The decision to use the RESIDENT option should be based on the frequency with which the PSB is used. If the frequency of use causes at least one copy of the PSB to normally be in the pool even if it is not defined as resident, it should be declared resident. If the PSB is used only occasionally, do not declare it as resident, so that its space in the PSB pool can be released when the PSB is not being used.

Related Reading: For more information on PSB options, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Using a Dynamic PSB

When you specify the dynamic PSB option, using the DOPT parameter on the APPLCTN macro, the PSB is read from the active ACBLIB library each time the program is scheduled. This allows an ACBGEN for a PSB to be performed while the online IMS system is running. The result of that ACBGEN can be reflected in the next scheduling of the program that uses that PSB.

DOPT can be applicable in a test environment or it can provide a PSB to be used with the Online Database Image Copy utility.

The following restrictions apply for a DOPT PSB:

- A dynamic PSB cannot be made resident.
- A program using a dynamic PSB cannot be scheduled in parallel with other programs.
- An MPP scheduled against a dynamic PSB is not eligible to go through quick reschedule or to become a pseudo WFI.
- All databases referenced by the dynamic PSB must have been defined to the system and be present in ACBLIB at system initialization or after the last online change was performed. This is because a corresponding option does not exist for DMBs, even though the PSB is dynamic.
- The current ACBLIB must consist of two or more concatenated data sets, and the dynamic PSB must reside in any data set in the concatenation other than the first. The concatenated data sets must be of the same format and contain the output from an ACBGEN.

Declaring Batch Message Programs

You must describe a BMP as batch on the PGMTYPE keyword. Each BMP is included as a separate application program with its own APPLCTN macro. You declare the BMP program by its use of a PSB, even though the batch JCL allows you to specify a program name that is different from the PSB name.

If a generalized BMP can execute with different PSBs, you must include the APPLCTN macros for all of them. A choice of PSB usually involves several queues that can be processed by the one program. At execution, you use the IN= parameter to match the transaction queue to the PSB.

Generated Program Specification Blocks

A generated program specification block (GPSB) allows the scheduling function of DB/DC, DBCTL, or DCCTL to generate an I/O PCB and a modifiable alternate PCB, each of which is used for the duration of a single application program execution, and terminated when the application program terminates. GPSBs eliminate the need to do PSB generation and ACB generation for a PSB that contains no database PCBs or alternate PCBs.

You specify GPSBs for an online environment using the GPSB= parameter on the APPLCTN system definition macro.

Restriction: GPSBs are not available in DB batch environments.

Related Reading: For instructions on implementing these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Declaring Fast Path Application Programs

You use the APPLCTN macro to declare the Fast Path application program name and whether it is a message-driven program. The PSB keyword is used to specify the PSB name, which is the same name as that of the message-driven program.

Declaring Program Processing

When you specify the PGMTYPE keyword value as TP, you designate the application program as message-driven. At the same time, you can specify FPATH=YES or FPATH=SIZE if you are certain the application program is always to be executed under Fast Path control. FPATH=SIZE establishes an Expedited Message Handler Buffer (EMHB) size for the application program. Specify FPATH=SIZE for the application program if its input or output message requirements are larger than the system default.

Related Reading: For more information on the FPATH parameter, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

One special effect of explicitly declaring the application program to be a message-driven Fast Path program is that the subsequent TRANSACT macro automatically specifies the transaction to be Fast Path exclusive. The sequence of APPLCTN and TRANSACT macros also causes a routing code, with the same value as the transaction code, to be automatically generated in a routing code table; the routing code indicates that this is for a Fast Path-exclusive transaction.

If you do not explicitly declare the application program to be Fast Path, the sequence of APPLCTN and TRANSACT macros (when FPATH=SIZE or FPATH=YES is specified on the TRANSACT macro) identifies a Fast Path-potential transaction.

Most of the other APPLCTN keywords have the same use as for DL/I application programs. However, for the PGMTYPE keyword, the value of BATCH is not used for Fast Path programs. Also, the OVLY parameter and the option to declare a message class are not valid for Fast Path application programs. Parallel scheduling is specified with the keyword value SCHDTYP=PARALLEL if the message-driven program is to occupy more than one Fast Path dependent region for a balancing group.

Adding Routing Codes

You use the RTCODE macro to declare any routing codes that are to be associated with the program specified in the preceding APPLCTN macro statement. You need

one RTCODE statement for each routing code. The name can be from 1 to 8 alphanumeric characters. It can be a duplicate of a transaction code or an LTERM name, but each routing code must be unique. These routing codes are in addition to those automatically generated by the APPLCTN-TRANSACT macro sequence.

You use the INQUIRY keyword to specify that, when this routing code is used to send a transaction to a balancing group, the program must use inquiry-only processing. INQUIRY=YES should be specified only for those transactions that do not cause a change to any database.

Defining IMS Transactions

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The TRANSACT macro influences the overall online IMS system response to incoming messages. You relate one or more transaction codes to the PSB named in the prior APPLCTN macro using the TRANSACT macro.

Related Reading:

- For more information on the transaction characteristics declared in the TRANSACT macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For the characteristics that govern the scheduling of the programs to process the transactions, see “Planning a Scheduling Algorithm” on page 62.

Analyzing Transaction Information

Analyze the application program to extract the details of how a transaction is to be processed and how the transaction looks to the end user. This information assists you when analyzing the specifications for transactions and determining the following:

- Specify the wait-for-input keyword, WFI, if a program must be continuously available and not reloaded each time it is scheduled.
- Specify YES as the first parameter of the INQ keyword if the transaction is only an inquiry and its database processing does not cause any updates. You probably do not require these to be recovered at system restart and can specify NORECOV as the second parameter of the INQ keyword.

  If update processing might occur, you must specify INQ=(NO,RECOVER) to request a recoverable transaction. When the current transaction processing is interrupted by an abnormal termination, the database changes are backed out and the input message is restored to the queue for reprocessing.
- When you are assessing input actions, determine if the response to an input message should be the next event rather than allowing multiple message input actions. The second parameter on the MSGTYPE keyword controls this response or no response:
  - The RESPONSE value inhibits further input.
  - The NONRESPONSE value allows multiple message input actions.

  The design of the transaction’s use of the input terminal determines the specifications for the MSGTYPE keyword. The expected input message characteristic can be a single segment, as indicated by an end-of-segment character, or it can be multiple segments. Scheduling for a multi-segment transaction cannot begin until the end-of-message character has been received.

- Specify the MODE keyword. Online processing works most efficiently with independent transactions specified as MODE=SNGL. Conversational or WFI transactions must be MODE=SNGL. For a dependent region to be eligible for quick reschedule or to become a pseudo WFI, transactions must be MODE=SNGL.

  If a group of transactions must be processed together, MODE=MULT ensures that database updates and output messages are kept together and are only committed to the database and output message queues when all of the transactions have been processed. The effect is that emergency restart processing backs out any database changes and reprocesses the whole group.
- Declare the transaction to be conversational by including the SPA keyword value; this value specifies the scratchpad area (SPA) size in bytes.
- Declare the presence of an input message edit routine with the EDIT keyword. Give the name of the routine as the member name of the link-edited module in IMS.USERLIB (or the equivalent library). The module must be present in the library before stage 2 of system definition processing. If the input is to be translated to uppercase before it is presented to the program, specify UC as the first subparameter value.
- Specify the parameters for output limits as a form of protection for your output message queues:
  - The SEGSIZE keyword causes a test of the size of an output segment.
  - The SEGEND keyword causes a test of the number of segments issued per transaction processed.

  These tests prevent exceptional output, probably in error, from entering the queues.

Application Program Output Limits

By establishing program output limits during system definition, you can safeguard the number and size of the output segments from the application program to the message queues. You use the SEGNO keyword to set the maximum number of output message segments allowed for each input message processed by the scheduled program. This gives you a way to protect available message queue space from being used up by a program output loop.

When an application program exceeds the output limits, a status code is returned indicating an error. Any further attempt by the application program to exceed the limits results in abnormal termination of the program. You can prevent abnormal terminations by having the program itself check the number and size of application program segments. This process of checking helps prevent IMS system abnormal terminations that occur when application programs loop while inserting messages or segments into the message queues, or when they inadvertently insert segments of invalid lengths.

Defining Fast Path Transactions

The TRANSACT macro is used to identify the transaction code and processing characteristics for a Fast Path transaction. The code is specified with the CODE keyword, and, even though the scheduling of Fast Path programs uses the routing code for its queue selection, the code must be a unique name among the set of transactions, LTERMs, and link names.

You use the FPATH keyword to signify that this transaction is Fast Path potential. You specify a parameter value of YES. You only need to do this when the preceding APPLCTN macro does not use a FPATH=SIZE parameter value. The FPATH operand is ignored if your preceding APPLCTN macro explicitly specifies a

message-driven application program. This parameter generates a routing code in a table identical to the transaction code and marks it as Fast Path potential.

The following keywords for the TRANSACT macro differ slightly in specification from DL/I transactions:

MSGTYPE Specifies a single- or multiple-segment message and its processing mode. Use parameter values SNGLSEG and RESPONSE, because Fast Path transactions must be single segment and response mode.

INQUIRY Specifies an inquiry transaction. You must use the default parameter value RECOVER, because these transactions are processed in the same way as other Fast Path transactions.

EDIT Specifies translation to uppercase and the presence of an input message edit routine. The latter option is not valid for Fast Path-exclusive transactions. For Fast Path-potential transactions, the edit routine is only invoked if the transaction is routed to IMS.

PROCLIM Specifies the maximum processing time. For message-driven programs, PROCLIM is the limit for one transaction and uses the real elapsed time, because the terminal is in response mode. The count parameter is ignored.

FPBUF Specifies the Expedited Message Handler Buffer (EMHB) size for transactions.

Planning a Scheduling Algorithm

The basic approach to controlling an online IMS system with loaded queues is to let the demand control the scheduling of programs into a reasonable number of message regions.

The strategy for defining a scheduling algorithm is explained in the following sections:

You specify a working set of BMP and message regions that can execute concurrently with the first parameter of the MAXREGN keyword on the IMSCTRL macro. Other dependent regions can be dynamically allocated by the MTO using the /START command. Up to 255 dependent regions (the maximum allowable

number permitted by IMS) can be allocated. For a DBCTL environment, the MAXREGN keyword defines a working set of BMPs and CCTL region threads, or application programs, which can execute concurrently. Other BMP regions can be dynamically started, up to the maximum allowable number, using the /START command. Recall that the DBCTL environment does not have any transactions, so further information about transactions in this section does not apply to DBCTL.

Related Reading: For more information on planning a scheduling algorithm, see "Assigning System Resource Options" on page 76.

Grouping Application Transactions

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

You must associate different groups of transactions with a particular message class that can be assigned to a region. (Batch message programs are allocated in their own region.) Criteria might be the virtual storage the message processing programs require, the PSB characteristics they share, or the priority of service for the end user. Lay out the transactions in a summary matrix. An example of this information summary for a vehicle routing application is shown in Table 5.

Table 5. Example of Transaction Grouping

Transaction Group	Tran. Code	Tran. Rate	Program	Preload	REGN	Processing Mode
Driver log	TRLOG	1000/day	PGMA	-	520 KB	SINGLE
Driver changes	TRCHG	300/hr	PGMB	PRLD	200 KB	MULTIPLE
Enter load/job	TRLOAD	50/hr	PGMC	-	520 KB	CONVERSATION
Status of job	TRSTAT	20/hr	PGMD	-	520 KB	SINGLE
Optimize route	TROPT	10/day	PGME	-	400 KB	(SINGLE) BMP
Get driver route	TRROUT	50/hr	PGMF	-	100 KB	SINGLE

One solution for this group of transactions is to define a message class for TRLOG, TRLOAD, and TRSTAT, which seem to be frequently used, and assign them to a message region. The BMP has its own region. The driver control transactions, TRCHG and TRROUT, can be assigned to another message class, because they have similar virtual storage requirements and one of them needs the program preload function specified on the region JCL.

The process of grouping transactions becomes more complex when you have competing application programs. An additional factor to consider is the database processing intent. You should attempt to combine compatible application programs (such as application programs that are inquiry only) into groups before assigning them a message class and individual priorities.

Assigning Message Class and Initializing a Region

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Each message (transaction code) is assigned a class using the third parameter of the MSGTYPE keyword on the TRANSACT macro. If not specified this way, the value on the PGMTYPE keyword of the APPLCTN macro is applied. This class

assignment determines into which message region an application program is loaded. When the IMS message regions are started, they are assigned from one to four message classes. When a message region is assigned more than one class, the scheduling algorithm treats the first class specified as the highest priority class, and each succeeding class is treated as a lower-priority class.

If more than one class is specified, message selection is processed as follows. The first class specified is scanned, in transaction-priority sequence, for waiting messages. If no messages are waiting for the first class, the second and following classes are also scanned in priority sequence. If messages are waiting in the first class, the highest-priority message is selected for scheduling.

Assigning Message Priorities within Message Class

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

To develop your scheduling algorithm, draw a matrix that lists each priority and transaction code. Group together all those transactions that belong to a class. You can then test the contents by setting up test levels of queue loading and apply the class and priority algorithms. Such a matrix is illustrated in Table 6.

Table 6. Matrix for Message Classes and Priorities

Class	Priority	Tran. Code	PSB Name
001	5	TRANX	PGMX
	3	TRAND	PGMD
	2	TRANA	PGMA
003	10	TRANY	PGMY
005	10	TRANC	PGMC
	5	TRANB	PGMB

If an available region specifies a message class priority of 1, 5, 3 and if two transactions are in each queue, PGMX is scheduled first and both TRANXs are processed. If no other transactions are received, the order of processing is TRAND, TRANA, TRANC, TRANB, and TRANY.

Notice how the order of message class prioritizing (specified in the region JCL) causes class 3 to be processed last.

If another message region specifying message classes 1, 5, 3 is started during the processing of TRAND, the region begins processing with PGMA. Whichever region completes message class 1 transactions first schedules PGMC.

Specifying Selection Priorities

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

When more than one transaction of a given type is waiting to be scheduled, the specified transaction scheduling priority determines which transaction code is selected. It does not determine which transaction is actually scheduled. Only the tests of the transaction's readiness for scheduling, which occur after selection,

determine if the transaction queue is allocated to an application program. The selection priorities are useful for influencing the response time to input transactions and for load balancing. Two priorities can be specified:

- Normal priority
- Limit priority

Related to the normal and limit priorities is the *limit count*. When the number of input messages of a specific transaction type waiting to be scheduled is equal to or greater than the limit count, the normal priority is reset to the limit priority value.

The priority of a transaction code causes it to be selected either before or after other transaction codes. You specify the numeric priority with the PRTY keyword of the TRANSACT macro. Values can be selected in the range 0 to 14; a value of 0 specifies the transaction is not eligible for automatic scheduling. If multiple transaction codes are at the same priority, they are selected on a first-in/first-out basis. However, if multiple transaction codes are at the same priority and the same class, with many messages already enqueued for each transaction code, the individual messages might not be selected in the same sequence in which they were entered.

For example, A, B, and C are transaction codes with processing limit counts of 1. These codes are entered in the sequence:

A1 B1 C1 B2 A2 C2 C3 A3 C4

The sequence in which they are selected is:

A1 B1 C1 A2 B2 C2 A3 C3 C4

You can raise the priority normally used for a transaction after a certain level of the queue is reached. In this way, you can give the transaction an increased chance of being scheduled. Another case occurs when a program requires significant program loading time or initialization and is then followed by a batch-like processing of a group of transactions.

Suppose the transaction TRANB in Table 6 on page 64 is assigned a limit priority of 14 if the number of queued transactions rises to 10. When message class 5 is available for scheduling and the queue counts for TRANC and TRANB are 18 and 10, respectively, the first program scheduled is PGMB. The processing of TRANB stays at priority 14 until all 10 transactions, and any others added to the queue, are processed. Then TRANB reverts back to a normal priority of 5.

It is possible that more messages will be added to the queue while the transaction is waiting or in process at the limit priority. The normal priority is not restored until all messages enqueued on the transaction code are processed. The priorities are selection priorities, not execution priorities. After a transaction has been selected for scheduling, the selection priorities have no influence until it is again recognized to be waiting for scheduling.

Limit priority can be in the range 0 to 14. Limit count has a default of 65535 and a valid range of 1 to 65535. You specify limit priority and the queue count as the second and third parameters of the PRTY keyword on the TRANSACT macro. If you do not require this priority override technique, code the limit priority equal to the normal priority, and code the limit count as 65535.

Another way to use the selection priorities is to declare a normal priority value of zero. Zero priority is a null or “not eligible for scheduling” level. Messages accumulate until the limit count is reached; at this point, limit priority takes effect and the message is eligible for scheduling. This technique is called *batching messages*.

The effectiveness of the selection priority assignments is related to how frequently the selection process occurs.

Setting Processing Limits

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

By setting processing limits, you can influence the frequency with which scheduling selection occurs. During the time between each scheduling, processing continues in the message regions. Meanwhile, messages are accumulating in the message queues. As messages accumulate, the interactive effects introduced by new message types and the changing of selection priorities are rearranging the order of waiting transaction codes. Conceivably, while a large queue of messages is being processed, important activity assigned to a high-priority transaction code is waiting.

When the program processes a large queue of messages and updates database segments, other application programs trying to access an updated segment are placed into a wait state. The length of time that the other application programs must wait depends on whether the updating program is processing its queue in multiple- or single-message mode.

To allow controlled reentry to the message scheduling selection process, specify a processing limit count for each transaction code. Each time a scheduled (processing) program requests a new message, the limit count is checked. When the number of requests exceeds the limit count, IMS determines if the region is eligible for quick reschedule.

- If the region is not eligible for quick reschedule, the application program completes its processing. If the current transaction code is at the same priority level as other queued transaction codes, the current code is placed last in the list.
- If the region is eligible for quick reschedule, the application program remains active and the next message is returned to the application program for processing.

Specifying Quick Reschedule

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Quick reschedule allows application programs to process more than the processing limit of messages for each physical schedule. Quick reschedule eliminates processing overhead caused by unnecessary rescheduling and reloading of application programs.

When a region undergoes quick reschedule, the message count that is compared with the processing limit count is reset, the accounting (X'07') and scheduling (X'08') log records are written, and the next message is returned to the application program for processing.

A region can undergo quick reschedule only when:

- No other work of equal or higher priority exists for the region to process
- The same transaction would be scheduled if the application program terminated and the dependent region went through rescheduling.
- The region is an MPP processing a MODE=SNGL transaction
- The processing limit count is greater than zero
- The PSB is not allocated with the dynamic PSB option (DOPT)

Flags in the accounting and scheduling records written during a quick reschedule indicate that the records do not include actual program termination and scheduling times. These records are written for accounting purposes only. Restart and backout do not use these records.

Specifying Pseudo WFI

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The pseudo WFI (pseudo wait-for-input) option allows an MPP region to remain scheduled until another input message appears. With pseudo WFI, unnecessary application program termination and rescheduling can be eliminated.

Normally, if an MPP region is scheduled for a transaction and no more messages for that transaction exist, the application program terminates. Frequently, another message appears for the same transaction after the program is terminated. Processing overhead is increased because of unnecessary termination and rescheduling of that application program.

Pseudo WFI is specified with the PWFI= parameter on the MPP region start-up procedure. When PWFI=Y is specified, the processing limit count is greater than 0, and no more messages are queued for the current MODE=SNGL transaction, IMS checks for other work for the region to process. If no other work is available, the region waits until another input message appears. This is *wait-for-input mode*.

When the next input message is for the currently scheduled transaction, the message is returned to the application program with a status code of “blank-blank”.

When the next message is not for the currently scheduled transaction, termination and rescheduling occur.

Certain circumstances cause regions that are in wait-for-input mode to be posted and a QC status code to be returned to the application program. These circumstances include commands that involve stopping, starting, locking, unlocking, purging, and assigning a database, region, transaction, or class. Regions that cannot be scheduled because of a lack of pool space can also post regions currently in pseudo WFI in an attempt to terminate them. This frees pool space so that the failing region can schedule.

Processing Transactions with Unavailable Data

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

IMS schedules an application program even if that application program might try to access an unavailable database. The application program can be *sensitive* or *insensitive* to unavailable data. To be sensitive, it must issue the INIT call. This requests that a status code be returned in the PSB if a subsequent call requires access to data that is unavailable. If the application program has not issued the INIT call and a call requires access to unavailable data, IMS abends the application program and backs out any updates it has made.

The disposition of the transaction depends on whether it is serial or not. Serial transactions are those that must be processed in the order of arrival. If it is serial, the processing of any transaction of its type is stopped. If it is not serial, only the processing of this particular transaction is stopped.

Data can be unavailable for these reasons:

- The database is stopped, locked, or unavailable for update.
- A lock cannot be obtained, because it is held by a failing component in a data sharing environment.
- In an XRF environment with block-level data sharing, the takeover system initiated new work before the data sharing configuration is revalidated.

IMS tries to resume transaction processing when any of these events occurs:

- A `/DEQ SUSPEND` command is issued.
- A `/START TRAN` command is issued for a transaction type that is stopped or that has stopped messages.
- A `/START DATABASE` command is issued for a database that is unavailable and in the intent list of the program requesting the transaction.
- A failing IRLM is reconnected.
- An emergency restart completes.
- An XRF takeover completes.
- A sharing IMS system completes a batch backout.

Scheduling Transactions Using the Suspend Queue

The operation of the suspend queue is documented by describing the specific elements of its operation. You must describe the following:

- The conditions that result in messages being placed in or removed from the suspend queue
- The conditions that result in setting or resetting the transaction stopped because of unavailable data (USTOPPED)

When Messages Are Placed in the Suspend Queue: If the program processing the message attempts to access data in a database that is unavailable, and the program has not issued the INIT call indicating that it can accept a status code that data is not available, the program is pseudoabended with abend U3303. The disposition of the message in process at the time of abend U3303 depends on whether the transaction type being processed requires serial processing. If the transaction type does not require serial processing, the failed message is placed in

the suspend queue. If the transaction requires serial processing, the message is returned to the normal queue and the transaction is USTOPPED.

Data might be unavailable for any of the following reasons:

- The database is stopped, locked, or not available for update.

  Programs are scheduled even when full-function databases are not available, or when they are available as read only. If a program issues a DL/I call that requires access to one of these databases, the program encounters unavailable data.
- A lock on the data cannot be obtained because it is held in retained state.

  In a block-level data sharing environment, it might not be possible to communicate with the sharing system because the sharing IMS has failed, the sharing IRLM has failed, or communication with the sharing IRLM has failed. The IRLM being used by the surviving IMS system retains knowledge of the locks that were held by the IMS system with which communication is temporarily unavailable. These locks are held in retained state. A similar condition can exist in a DBCTL environment when a thread failure occurs.
- In an XRF and block-level data sharing environment, the takeover system initiates new work before the data sharing configuration is revalidated.

  At the time of an XRF takeover, databases that can be shared at the block level are temporarily made unavailable until the data sharing configuration has been revalidated. If programs attempt to access these databases before the revalidation completes, they encounter unavailable data.

When Messages Are Removed from the Suspend Queue: A separate suspend queue exists for each transaction type. Messages are never scheduled for processing from the suspend queue. To be scheduled, the message must be transferred to its normal queue. Some conditions cause the messages on suspend queues for all transaction types to be transferred to their normal queues. Other conditions cause the messages for specific transaction types to be transferred to their normal queue.

The conditions that trigger the transfer of all messages from the suspend queues, and the rationale for transferring the messages when that condition occurs, are:

- The `/DEQ SUSPEND` command is issued.

  The operator requested it.
- IMS emergency restart is completed.

  While the IMS system is down, a sharing IMS might notify the system to drain its suspend queues.
- A sharing IMS system notifies the system that the sharing IMS system has completed an emergency restart or a batch backout.

  Messages are transferred for the same reason as when these conditions occur on the local system.
- IRLM is reconnected.

  When the IRLM failed, messages that were in process at the time of failure were abended with abend U3303. Attempts to access data that the failing IRLM locked also result in abend U3303. When the IRLM is reconnected, these messages are scheduled again.
- XRF takeover has completed.

  While the takeover is in process, a notification from a sharing IMS might have been missed, and databases that can be shared at the block level are temporarily unavailable until the reverification to DBRC has completed.

The following conditions trigger the transfer of messages for specific transactions to the normal queue:

- A /START TRAN command is issued. This causes the messages for the started transaction to be transferred to the normal queue from the suspend queue.
- A /START DATABASE command is issued. This causes the transfer of messages for transactions in which the program processing the transaction has access to the started database.

Setting the Transaction USTOPPED Because of Unavailable Data: IMS stops the transaction type if most messages being processed are failing because they are encountering unavailable data. One abort is counted each time the program aborts due to abend U3303 and the message in process has not previously been placed on the suspend queue. Two aborts are subtracted each time a program goes through commit processing. However, if this results in a negative number, two aborts are not subtracted. If the total number of aborts exceeds 10, the transaction is stopped via a USTOPPED condition.

The transaction is stopped by USTOPPED if abend U3303 occurs while processing the message and SERIAL=YES is specified for the TRANSACT macro.

When the transaction is stopped by USTOPPED, messages from this queue are not scheduled for processing. Incoming messages continue to be queued on the normal queue.

Resetting the Transaction Stopped Because of Unavailable Data: The USTOPPED condition is reset under the following conditions:

- A /START DATABASE command resets the USTOPPED condition for all transaction types when the program that processes this transaction has access to the started database.
- A /START TRAN command resets the USTOPPED condition for that transaction type.
- A /PURGE TRAN command resets the USTOPPED condition for that transaction type.

Specifying Parallel Scheduling

IMS can schedule the same application program and the same transaction in multiple message regions. Designate the application program and the transaction for parallel scheduling using the SCHDTYP keyword on the APPLCTN macro. You must designate the application program as a parallel-scheduled application program so that any transaction processed by that program can be scheduled in multiple regions.

When a transaction is available for scheduling but is already scheduled in another region, IMS checks whether the transaction can be scheduled in parallel. The PARLIM value of the TRANSACT macro specifies the number of messages that should be enqueued before another region is scheduled. This value is multiplied by the number of regions already scheduled for this transaction. If the result is less than the number of messages enqueued, another region is scheduled for the transaction unless MAXRGN is exceeded. If the region cannot be scheduled for internal reasons (database intent), the next transaction within the class is scheduled.

If scheduling fails for intent conflicts for a SCHD=1 or SCHD=2 transaction, the next logical transaction is selected based on the SCHD= parameter of the transaction that failed. If the scheduler fails to schedule any transaction for intent 5 times, the next class of transactions is selected.

This method of processing can cause messages in the current class to be delayed. For example, a long running BMP has update intents on a database. Several transactions that have update intent on the same database with SCHD=1 are entered into the IMS system. More transactions that do not reference the database but have the same class are also entered into the system. These transactions might not be scheduled until the BMP terminates, if the first group of transactions fails for intent. When transactions from the first group fail for intent a total of 5 times, scheduling is attempted for the next class. This bypasses the group of transactions that do not reference the database.

To avoid such delays, the second group of transactions should be placed into a separate class, or the BMP job should be run at a different time.

If the PARLIM value is zero and more messages are in the queue, another region can be scheduled. To prevent one transaction from monopolizing all available regions, use the MAXRGN= parameter on the TRANSACT macro. A non-zero value for MAXRGN specifies the number of MPP regions that can be scheduled. In addition, you can use the SERIAL option of the TRANSACT macro to process transactions in the order they arrive. IMS limits the processing to this time sequence. If data required by the transaction is unavailable, this causes IMS to stop scheduling this transaction type.

For a DBCTL environment, BMP regions and CCTL threads can schedule a PSB simultaneously when the APPLCTN macro's SCHDTYP keyword is defined as PARALLEL. If a PSB is not defined as PARALLEL and is already scheduled by a BMP or CCTL thread, new schedule requests for that PSB fail.

Specifying Alternative Scheduling Options

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

If a message cannot be scheduled for external reasons (for example, program or transaction stopped by the master terminal operator), the next message of equal or lower priority in that class, or the highest-priority message in a lower class, is selected for scheduling. If the highest-priority message in the first class cannot be scheduled for internal reasons (database intent, no more space in PSB pool, or DMB pool to bring in needed blocks), the scheduling option of the transaction specifies the criteria to be used to select the next transaction to be scheduled. The SCHD keyword of the option is specified at system definition by the TRANSACT macro. The options are:

- Schedule only transactions of equal or higher priority in the selected class. This is the default option.
- Schedule higher-priority transactions in the selected class.
- Schedule any transaction in the selected class.
- Skip to the next class and attempt to schedule the highest-priority transaction in that class.

These scheduling options are specified for each transaction; therefore, if the algorithms are different for transactions within the same class, each attempt to schedule a different transaction might change the algorithm.

Message region class assignments and transaction class assignments are assigned at region initialization through the EXEC JCL statement. The assignments can be modified at execution time by the operator.

If multiple message regions process the same message class and a conflict in database processing intent occurs, the highest-priority transactions scheduled against a database are not necessarily processed before lower-priority transactions scheduled against the same database. If you want to process all higher-priority transactions before processing any lower-priority transactions, specify no processing limit for the higher-priority transactions. Using only one message region to process that message class achieves the same result.

Scheduling for BMP Processing

Because BMP regions are scheduled manually, the input transactions to a batch message program do not need to be assigned a competitive priority. Specify a priority value of zero for both normal and limit priority, and coordinate a message class designation with the BMP region JCL.

Assigning Priorities for Programs with Exclusive Intent

When a program's PSB includes exclusive use of a segment type, the Program is not scheduled concurrently with any other program that is sensitive to the same segment type. Similarly, when a program executes with exclusive use of segment types, other programs that include sensitivity to any of those segments are not scheduled concurrently. Exclusive intent does not use enqueue/dequeue serialization. (Programs have exclusive intent declared by PROCOPT=E on a SENSEG or program communication block, or PCB, statement within the PSB generation—if the option K, for key sensitivity, is not appended.) Conflicting actions occur only if the same segment type is declared by at least one of two programs intending to reference a segment exclusively.

One case in which programs require exclusive intent occurs when a program that uses HSAM in its PSB is scheduled.

Use care when assigning priorities for programs with exclusive intent. Even if a program is selected for execution, a conflict with its processing intent and that of an already-executing program causes the transaction to drop out of the selection process until the next program termination or region start event.

Exceptions to the use of program isolation are programs that use the PROCOPT=GO option. These programs can retrieve segments that have been altered or modified by programs that are still active. Those changes might be subject to backout. The programs might not update the segments, and there is no enqueue on the segments when the programs retrieve them.

Scheduling for CPI-Communications-Driven Programs

Scheduling information for LU 6.2 CPI-Communications-driven application programs is defined in a TP_Profile entry managed by APPC/MVS. When APPC/IMS recognizes a CPI-Communications-driven application program for the first time after restart, it dynamically builds an IMS transaction. IMS dynamically builds the definition for CPI-Communications-driven application programs when a transaction is presented for scheduling by APPC/MVS, based on the APPC/MVS TP_Profile definition after IMS restart. A dynamically-built transaction is not checkpointed unless SYNCLVL=SYNCPT and IMS is participating in a protected conversation.

Related Reading: For more information on CPI-Communications-driven application programs, see *IMS/ESA Administration Guide: Transaction Manager*.

Defining IMS Terminals

Most of the system definition stage 1 input is made up of declarations that define terminals to be attached to the online IMS system. You can use the detailed information collected in the terminal profiles and configuration diagrams for this information. Each terminal type and hardware option has its equivalent parameter in one or more of the terminal-related macros in the stage 1 input. Four sets of macros are defined in the following order:

1. BTAM-, BSAM-, GAM-, and ARAM-supported devices
2. Switched communication devices
3. MSC communication links
4. VTAM-supported terminals

VTAM terminals can be introduced in three ways:

- At system definition, using a set of macro statements
- At any time, using the Extended Terminal Option (ETO)
- At any time, using LU 6.2

Related Reading: For information on the attributes specified for Multiple Systems Coupling (MSC), see *IMS/ESA Administration Guide: Transaction Manager*.

Defining Extended Terminal Option Terminals

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

ETO allows VTAM terminals and LTERMs (logical terminal users) to be added to IMS without redefining the system. ETO must be installed with the IMSCTRL macro and activated using the ETO=YES execution parameter. However, you can still override the initialization of ETO by using the Initialization exit routine (DFSINTX0).

Related Reading: For more information on this exit routine, see *IMS/ESA Customization Guide*.

ETO descriptors are templates that provide information about:

- Physical characteristics of terminals
- User options and message queue (LTERM) names
- Remote LTERM locations associated with MSC links

An optional starter set of ETO descriptors can be provided at system definition. You can begin with the starter set and expand on it as necessary. You can also install ETO exit routines that allow IMS to dynamically create LTERMs, even when the specified descriptors do not contain LTERM-created data.

Terminals defined with ETO are described with parameters similar to system-defined terminals. However, several keywords for these parameters are not applicable to ETO, and must not be specified. In addition, the master terminal and the XRF surveillance link cannot be defined with ETO.

Related Reading: For more information on ETO, see *IMS/ESA Administration Guide: Transaction Manager*.

Defining Static VTAM Terminals at System Definition

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

In a set of macro statements, you specify all the VTAM-supported terminals that operate in the IMS online system.

Related Reading: For more information on specifying VTAM-supported terminals, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Defining Non-VTAM Terminals

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

You specify all BTAM, BSAM, or ARAM devices that are to operate in the IMS online system in a set of macro statements. You follow this with a set of statements that describe all switched devices.

Related Reading: For more information on specifying non-VTAM terminals, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Specifying the Master Terminal

The key control point for IMS online operations is the master terminal. You will probably choose a screen device because of the advantages of convenient data entry and output response. However, you need printed copy of many of the responses to commands, as well as a record of the system messages sent to the master terminal. You can specify a secondary master terminal to use for print output.

You can also specify secondary system consoles for database operator communication within DBCTL. Consoles with the proper matching characteristics receive unsolicited messages from the system.

Related Reading: For more information on specifying secondary system consoles, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Choosing Master Terminal Devices

Some restrictions exist in the device type for primary and secondary master terminals. Your choice for secondary master depends on the expected amount of output and the promptness of printing. Primary and secondary master terminals cannot be defined with ETO or LU 6.2. In addition, FINANCE, LUP, and ISC terminals cannot be master terminals.

The `/ASSIGN` command can be used to switch the secondary master console to another destination such as a spool SYSOUT line group.

Choosing the Extent of Secondary Master Logging

To provide for automatic copying of the entry and response of key system control commands, you can specify, on the COMM macro, values for the COPYLOG keyword to cause the command activity to be copied when issued by the master terminal (or by any terminal). Your safest choice is to specify ALL, unless terminals

other than the master terminal will be issuing many commands. The choice of NONE or NOMASTER is not recommended, because the printed log of activity provides a valuable audit mechanism.

Defining Switched Devices

The following sections highlight the use of special system definition macros when you plan to use switched devices in your IMS network.

Related Reading: For examples of the macro coding and representative configurations, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Defining Switched 3275 Devices

If you plan to use switched 3275 devices, you must specify the configuration. Choose a line group and follow the LINEGRP data with details of the hardware options indicated by CONFIG macro keywords. 3275 hardware includes a security option in that each 3275 carries a physical identification number as part of its circuitry. The last three bytes, together with a unique configuration name (the label on the CONFIG macro), constitute the IMS identification. List all authorized identifications for a line group on an IDLIST macro. Subsequent LINE macros coded for 3275 devices can point to these lists.

Defining LTERM Names for Switched Devices

If you have switched communication devices in your requirements, you relate them to a previously defined line group and include the macro specifications POOL, USER, and NAME.

Related Reading: For more information on these specifications, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Defining System/3 and System/7 Stations

If you plan to include remote intelligent stations in the IMS network, you must specify the station's physical and logical characteristics on a STATION macro. Set aside one or more line groups, and then add the characteristics of one or more System/3 or System/7 stations. A station is primarily identified by its assigned LTERM name. IMS assigns default LTERM names as it encounters STATION macros—RSTSnnnn, where *nnnn* is a sequence number incremented from 0001.

The specifications for the LINEGRP macro allow you to declare a System/7 line to operate as a polled line or in contention mode. You also declare whether the transmission is to be binary synchronous or start-stop.

The terminals that are attached to a System/3 or System/7 are able to transmit messages through to IMS. However, the transmission block formats and protocols have to coincide with those defined for IMS intelligent remote station support (IRSS). Each terminal is assigned an LTERM name and can be restricted to input data only. Conversational processing as well as preset output destination is available.

Related Reading: For more information on how to communicate between IMS and IRSS, see *IMS/ESA Customization Guide*.

Allocating Message Format Pool Space

The IMS online system reserves storage for the message format pool. The amount of storage required depends primarily upon how much concurrent use of message format blocks is expected and how many lines are active.

Defining Pool Space for MFS Devices

Using the BUFPOOLS macro, you can control the size of the message format pool and the number of fetch request elements (FRE). One FRE is required to control each active block; without a FRE, space cannot be assigned from the message format pool. In estimating the number of FREs, be sure to add 10 or more to allow for system message activity. This is especially true if you anticipate status inquiries (`/DISPLAY` command) being made by MFS-supported terminal operators and do not want to have the possibility of delayed response to other terminals. If pre-allocated FREs are not available, a dynamic FRE is allocated from the general area of the pool. However, you should avoid using dynamic FREs, because they cause fragmentation of the pool.

For the message format pool, compile a list of the MFS blocks, arranging them in DIF/MID and DOF/MOD pairs. Next, record the sizes and whether they are used according to their priority, their quick-response transactions, or their frequency. Allow space for input/output pairs to be in the pool. Watch for large messages or those with multiple segments, because these can preempt small, frequent messages from finding space for their format blocks in the pool. This situation directly affects the response time of the small messages.

If your IMS system is generated for OS/390, the message format pool resides in extended private storage.

Message Format Indexing

You can use the MFS utility to build an index of the DASD addresses; the index resides in the message format pool during online execution. Building the index saves an I/O to the active IMS.FORMATA/B library directory to look up the physical address of the required block. (Each index entry requires 14 bytes.) You might not want to index all format blocks if they are especially numerous, but you should plan to index those that are frequently referenced. You specify selected index entries and the building of an index as part of the input control statements to the MFS utility.

Related Reading: For more information on building this index, see *IMS/ESA Utilities Reference: Transaction Manager*.

The index entries become part of an MFS dynamic directory that resides in extended private storage. Entries are added, if not found in the current index. The directory can be reset to its state at initialization by using an option provided with the `/CHANGE` command.

Assigning System Resource Options

A group of parameters within the system environment macro set relates to the integrity of the online IMS system. These parameters specify resources available for checkpoints and program isolation. The allocation of system data sets and control program storage depends on the expected processing load and the strategy for recovery of the online IMS system.

This section describes the following information:

• “Security Options” on page 82

Choosing the Number of Regions

Overall throughput is conditional on the number of active dependent regions and the availability of shared resources for database processing. A processing program must have an available message region before it can enter the scheduling selection logic.

You specify a working set of BMP and message regions with the MAXREGN keyword on the IMSCTRL macro. Additional dependent regions (up to the maximum of 255) can be dynamically allocated by the MTO using the /START command. CCTL threads can be allocated up to the number specified at system definition.

Related Reading: For more information on allocating CCTL threads, see *IMS/ESA Customization Guide*.

Your operations procedure stipulates how many, and when, regions are active. The scheduling algorithm controls the actual programs that execute in the regions in response to enqueued messages.

Defining a Fast DB Recovery Region

The Fast DB Recovery procedure executes a Fast DB Recovery region. The Fast DB Recovery procedure is similar to those used to define IMS subsystems (for example, the DBC procedure).

The FDRMBR parameter identifies the Fast DB Recovery region to IMS.PROCLIB with a two-digit suffix (FDRMBR=xx). The IMS.PROCLIB member is DFSFDRxx.

The following parameters are valid for the Fast DB Recovery procedure:

BSIZ	CSAPSB	DBBF	DBWP
DLIPSB	DMB	FDRMBR	GRNAME
IMSID	IRLMNM	LGNR	PSB
PSBW	SPM	SUF	UHASH
VSPEC	WADS	WKAP	

Exception: All of the valid parameters have the same definition for Fast DB Recovery as they have for the IMS and DBC procedures, except CSAPSB and DLIPSB. When the Fast DB Recovery procedure specifies CSAPSB and DLIPSB, the sum of their values defines the PSB pool size. When PSB is also specified, the larger value (PSB or the sum of CSAPSB and DLIPSB) is used.

Enabling a DB/DC Subsystem for Fast DB Recovery

To enable a DB/DC subsystem for Fast DB Recovery, specify the FDRMBR parameter in the IMS procedure. The FDRMBR parameter defines the DB/DC system as Fast DB Recovery-capable.

Note the IMS ID of the system that Fast DB Recovery is to track. Specify this ID in the control statement for the DFSFDRxx IMS.PROCLIB member. Match the IMS ID to the ID that is specified for the IMSID EXEC parameter.

Restriction: If both FDRMBR and HSBID (XRF configuration) parameters are specified in the DBC procedure, the FDRMBR parameter is ignored.

Enabling a DBCTL Subsystem for Fast DB Recovery

To enable a DBCTL subsystem for Fast DB Recovery, specify the FDRMBR parameter in the DBC procedure. The FDRMBR parameter defines the DBCTL system as Fast DB Recovery-capable.

Note the IMS ID of the system that Fast DB Recovery is to track. Specify this ID in the control statement for the DFSFDRxx IMS.PROCLIB member. Match the IMS ID to the ID that is specified for the IMSID EXEC parameter.

Restriction: If both FDRMBR and DBRSE (DBCTL standby configuration) parameters are specified in the DBC procedure, the FDRMBR parameter is ignored.

Setting a Checkpoint Frequency

The primary tool that IMS uses to record information for restarting interrupted operation is checkpointing. Using the status information captured during checkpoint, the content of the message queues and database changes can be restored. Checkpoints are an integral part of system shutdown and startup. The amount of reprocessing, back from the point of system interruption and forward to a continuation point, is minimized when checkpointing is reasonably frequent. Your trade-off is between efficient restart and processing overhead for the checkpoint information.

It is necessary to begin a restart at the current checkpoint minus one. This applies to restarts from all checkpoints except for RESTART, SNAPQ, and SHUTDOWN. For this reason, it is strongly recommended that you review your checkpoint frequency and perhaps take more frequent checkpoints in order to minimize the amount of time it takes to do the restart.

System service interruptions that can be caused by extensive checkpoint processing have been minimized. This situation makes it possible for you to take more frequent checkpoints and, at the same time, enhance restart performance.

You control the frequency of IMS internal checkpoints with the CPLOG keyword on the IMSCTF macro. The decision to invoke a checkpoint is based on an increment to the number of system log records created. You specify a number of records; the default is 1000. As the online IMS events are logged with individual record types, a count is maintained. When the increment exceeds the CPLOG value, checkpoint processing is invoked. IMS system checkpoints can also be invoked explicitly by the master terminal operator and by application programs that have been authorized to enter the `/CHECKPOINT` command.

Your interval should be large enough to extend beyond the estimated response time for at least one of the longest-running transactions.

You can adjust the frequency after observing the stability of the online IMS system. You can use the IMS Monitor reports to assess the processing overhead. The Region Summary report shows the total elapsed time and average elapsed time taken for checkpoints in the trace interval.

Selecting an IMS Lock Manager

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

To protect database integrity, an IMS system serializes requests for database resources so that application programs are prevented from updating a database segment until a current owner of that segment has indicated that any changes it made are completed. The process of controlling concurrent requests is called lock management. Choose one of two managers to control the stored information about the requests: a program isolation lock manager or an Internal Resource Lock Manager (IRLM) component. The program isolation lock manager can control lock requests for only a single IMS system, which is called local locking. The IRLM can control lock requests both for multiple IMS systems, which is called global locking, and for single IMS systems. The IRLM is required if the IMS online system is to take part in block-level sharing.

If you select program isolation to manage locks, you have the advantage of not needing to plan any special operating and recovery procedures, as is necessary for the IRLM. Also, monitoring program isolation activity is an integral part of the IMS Monitor, invoked with the `/TRACE` command by the MTO. There is also a separate program isolation trace that can be used in performance analysis. System definition requirements related to program isolation locking are explained in the next section.

Your choice of lock manager is not necessarily fixed. Using parameters on the EXEC statement for the control region, you can override the use of program isolation locking. In this way, you can allow for the IMS online system to take part in block-level data sharing. For details of how to initialize and plan operating procedures for an IRLM, see “Chapter 10. Administering a Data Sharing Environment” on page 305.

Even if you plan to use only IRLM for lock management, you must specify a minimal amount of enqueue/dequeue storage (a maximum of two isolated locks for each partition specification table, or PST) as described in the next section. IMS uses this storage internally.

Specifying Enqueue/Dequeue Requirements

The online IMS system protects the integrity of the database when concurrently running programs are updating the same database record. A record of the inter-level update events is maintained for program isolation in tables defined in the control program storage. These are termed enqueue/dequeue tables and consist of 24-byte entries in an OS/390 system. As programs reach synchronization points, the entries are freed and the storage can be reused. You need to estimate the number of events that might be recorded for concurrent execution of programs that might process against the same segment type.

The storage is allocated above the 16 MB line.

Selecting the DL/I Separate Address Space

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

You have the option of using a DL/I separate address space (DLISAS) to contain code, control blocks, and buffers for full-function databases. You do this by using the local storage option (LSO). You specify LSO=S as an EXEC parameter for the control region. Throughout this section, the term LSO=S is used to indicate the choice of a DL/I address space.

For DBCTL, the DLISAS is required. Therefore, the LSO parameter is not used. For a DB/DC environment running a DBCTL function, you must specify LSO=S.

The IMS control program automatically initiates the DL/I address space. If either the control or DL/I address space terminates, the other is automatically terminated.

MVS cross-memory services are used to process an application program database call. Because of the frequency with which cross memory operations are used, appropriate hardware support is recommended.

IMS restart procedures are insensitive to the LSO specification. For example, an LSO=S system can be terminated and IMS restart procedures can be performed on a system specifying LSO=Y.

DLISAS Procedure Modifications

You need to be aware of the following possible modifications required to the DL/I address space procedure:

- Ensure that the JCL DD statements for the full-function databases are in the DL/I address space procedure and not in the IMS procedure. JCL DD statements for Fast Path databases and the IMS system data sets remain in the IMS procedure.
- You do not need to make changes to the dynamic allocation parameter lists in IMS.RESLIB.
- Ensure that the specification of the active and inactive ACBLIBs are identical in both the IMS procedure and the DLISAS procedure. (Both the control region and the DL/I address space read ACBLIB.) The data sets used must have disposition SHR, and the concatenation order must be identical.
- Ensure that IMS.PROCLIB is defined in both the IMS and DLISAS procedures.
- For the DLISAS procedure, you only need to pass the region type (DLS) parameter and, optionally, the IMSID of the control program to which you want to connect. When IMSID is specified on startup parameters, the IMSID is passed on to the DBRC and DLISAS started tasks. Pool sizes and database buffering options are specified on the IMS start procedure.

Related Reading: For more information on DLISAS, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DL/I Exit Routine Modifications

In an LSO=S system, the DL/I exit routines are entered in cross-memory mode and certain operating system restrictions apply to this environment:

- The DL/I exit routines cannot address storage in the control address space.
- If the DL/I exit routines use the IMS ISWITCH service, they must be changed to function correctly in an LSO=S system. On ISWITCH, you must give the macro specification TO=DLI, not TO=CTL. The TO=DLI specification functions correctly in all IMS environments.

Storage Considerations

The LSO=S option moves the following major storage to the DLISAS private area:

- DL/I code
- Database buffers
- DMB pool, both resident and nonresident
- DMB work pool
- Most of the PSB pool, both resident and nonresident
- PI ENQ/DEQ tables for non-Fast Path systems

For an LSO=S system, the following storage is in the MVS common area:

- OSAM code

- Resident intent lists

The resident intent lists are in the control region private area for LSO=Y.

PSB Pool Definition

In an LSO=S system, two PSB pools exist: one in the MVS common area (DLMP) and one in the DL/I region private area (DPSB). The output of the ACBGEN utility indicates, for a given PSB, the amount of space required in each pool.

The sizes of these pools are specified with the SASPSB parameter on the BUFPOOLS system definition macro. You can override these sizes with the CSAPSB and DLIPSB parameters on the procedure used to start the online IMS system.

If LSO=S is not being used, a single PSB pool exists in the MVS common area. In this case, the PSB parameter on BUFPOOLS and on the IMS start procedure is used to specify the size of this pool.

The DPSB pool contains the DL/I control blocks associated with a full-function database PCB (for example, the JCB or Level Table). The DLMP pool contains the communications PCBs, Fast Path PCBs, and the key feedback area of the full-function PCBs.

The ACBGEN utility organizes the PSB so that all data for the DLMP pool precedes all data for the DPSB pool. This new organization applies to all IMS environments.

Other Planning Considerations

Accounting Procedures with an LSO=Y System: Accounting procedures based on the IMS system log, Resource Measurement Facility (RMF), and System Management Facility (SMF) are affected when LSO=Y is specified since DL/I time is counted under the IMS control region address space. The CPUTIME value recorded in the IMS system log is affected by the LSO option specified. If LSO=Y is specified, only application program processing time is recorded in the type07 log record; most DL/I processing time is not included.

For some applications, DL/I processing time can represent a large percentage of the total processing time. If LSO=S is specified, DL/I processing time is included in CPUTIME. Because the IMS system log does not indicate which LSO option is specified, CPUTIME values should not be used for accounting or comparison purposes when switching options.

Security Considerations: If the full-function databases are RACF protected, the DLISAS procedure must be authorized to access these resources.

Tuning Considerations: Tuning procedures must allow for the fact that the IMS control program consists of more than one address space.

The output of the ACBGEN utility indicates, for a given PSB, the amount of space required in each PSB pool. An out-of-space condition in the DPSB pool or a wait-for-storage condition is reflected in output reports of the IMS Monitor. In these reports, DLMP and DPSB represent the CSA PSB pool and the DLS PSB pool, respectively.

The command `/DISPLAY POOL PSBP` displays usage information for both PSB pools in an LSO=S system.

System Resource Options

To page fix the PSB pool, specify POOLS=DLMP and POOLS=DPSB in the DFSFIXxx member in IMS.PROCLIB. To fix the resident PSB pool, the module names DFSPSBRS and DFSDLIRS should be included in this member.

Security Options

The IMS system has several ways to prevent unauthorized use of a terminal or connected device known to the system. You can also control what processing is allowed in a dependent region. The use of resources is authorized by declaring the names of the IMS resources, such as the LTERM name or transaction code, in the input to the Security Maintenance utility, RACF (or equivalent product), and exit routines.

Note: Security Maintenance utility does not apply to ETO or LU 6.2 devices. Security for ETO and LU 6.2 devices is maintained with RACF and exit routines.

Different declarations can be made depending on the IMS security function selected. The type of security declaration has its counterpart in system definition as a value for the TYPE keyword for the SECURITY macro. The parameter values to invoke security checking are SIGNEXIT, TRANEXIT, and AGNEXIT. If you plan to use the RACF program product for your MVS system, you specify RACFCOM, RACFTERM, or RACFAGN. For several of the parameters you can specify values that determine whether the master terminal operator can override the security matrix status at restart time. These same keywords define the flexibility the MTO possesses for invoking or overriding the type of security checking that is to be active during the current online execution cycle.

The specification of the security options needs to be a part of the overall security design. For a description of design and operational considerations for security with IMS, See “Chapter 4. Establishing IMS Security” on page 115.

Initializing IMS System Data Sets

When installing a DB/DC environment, you must allocate and catalog IMS system libraries and online data sets. Although this activity is done independently of the actual system definition processing, you need to coordinate this activity with several parts of system definition stage 1 input.

Planning for the definition of IMS data sets requires you to predict the direct access storage to be allocated based on your anticipated application program workload. Some of the data sets you estimate are affected by the design decisions made for system definition. Most of these data sets are automatically generated as DD statements within the members of IMS.PROCLIB.

Related Reading: For more information on the data set allocation, see *IMS/ESA Installation Volume 1: Installation and Verification*.

Administering IMS System Data Sets for Online Change Function

In many installations, it is important that the online system be available to users during a large portion of the day. Many users want to move toward continuous operation of their IMS system. Users that require their online IMS system to be available continuously need to be able to modify that system online. The ability to add, delete, and replace IMS databases, programs, transactions, and MFS formats online without having to bring down your IMS system is a major step toward continuous operations.

For a DBCTL environment, no MFS facility exists, and the TRANSACT and RTCODE macros do not apply. A DCCTL environment has no database facilities. Therefore, the DATABASE keyword does not apply. When designing a DBCTL or DCCTL environment, use the following information as it applies to your system.

Adding, deleting, or changing the IMS resources involves changes to the control blocks set up for these resources. If your system uses the online change facility of IMS, it can require a special kind of system definition, a MODBLKS generation. Within this system definition, you specify appropriate changes to keyword parameters on the DATABASE, APPLCTN, TRANSACT, and RTCODE macro statements. A MODBLKS system definition generates the control block members for resources that are to be added or changed online. These control blocks, stored in the library IMS.MODBLKS, are used by the IMS control region, the Security Maintenance utility, and the MSC Verification utility when an online change to your IMS system is requested.

When installing the IMS online change function, you must create three copies of each of the following libraries:

- IMS.MODBLKS—the library that contains the control blocks to support online change of databases, programs, transactions, and MFS formats
- IMS.MATRIX—the library that contains your system's security tables
- IMS.ACBLIB—the library that contains database and program descriptors
- IMS.FORMAT—the library that contains your MFS maps produced by the MFS Language and Service utilities

These libraries are for the exclusive use of IMS offline functions and are called the *staging libraries*. For each library, a copy is made to produce a data set with a data set name suffixed with an A and a B, for example, IMS.FORMATA and IMS.FORMATB. These two copies of each library are used by the IMS online system.

At initial installation, the staging libraries and the IMS A libraries are identical. At this time, the A libraries are referred to as the *active libraries*. They are the libraries from which IMS draws its execution information. The B libraries are not used at this time and are referred to as the *inactive* libraries.

The online change function libraries work as follows:

1. You apply changes to the staging libraries.
2. The staging libraries are subsequently copied to the inactive (B) libraries using the Online Change utility.
3. Operator commands are issued to cause the B libraries to become the active ones; the old active (A) libraries become the inactive ones.

When you wish to add, replace, or delete any of the IMS resources previously mentioned, you apply your changes to the offline staging libraries by running:

- A MODBLKS system definition—if you have added, changed, or deleted applications, programs, full-function databases, or routing codes
- An ACBGEN—if you have added or changed any databases or programs
- The MFS Language and Service utilities—if you have added or changed any MFS format definitions
- The Security Maintenance utility—if you have added, changed, or deleted resources

Initializing Data Sets

You can apply changes to IMS.FORMAT, IMS.ACBLIB, or IMS.MATRIX independently or in combination. IMS.MODBLKS is changed by the MODBLKS system definition. If the security tables are changed, the suffix of the inactive library must match that of the inactive IMS.MODBLKS library.

After issuing the sequence of commands (`/MODIFY`) to cause the previously inactive libraries to become the active libraries, your previously active libraries become the inactive libraries. They are not destroyed until overwritten by the next online change sequence. This has the advantage of permitting you to return to this set of libraries if backup and recovery are necessary or if an incorrect definition occurs during your online change run. Additionally, IMS monitors for you which set of libraries is currently active. This information is kept in a status data set, IMS.MODSTAT.

After an online change is successfully completed, it persists across all types of IMS restarts. Additionally, the new resources can be easily maintained by running an SMP JCLIN against the stage 1 output stream produced by your MODBLKS system definition to record the contents of the new system definition in your SMP control data set. This ensures that any maintenance applied to your IMS system is applied to the currently active system.

IMS Online Data Sets

The online IMS data sets you need to define are shown in Table 7.

Table 7. List of Online IMS Data Sets

ddname	Data Set/Content	Prerequisite
IEFRDER	IMS.JOBS (PDS)	Use of IMSRDR to start regions
STEPLIB	IMS.RESLIB (PDS)	APF library authorization
DFSESL	Optional use for external subsystems	APF library authorization
PROCLIB	IMS.PROCLIB (PDS)	Procedures and initialization, use of DL/I address space
MODBLKSA/B	From staging IMS.MODBLKS	Modified for online change
MODSTAT	Active library list	INITMOD procedure, Online Change utility
DFSOLPnn[1]	Online log data set (primary)	Archiving, dynamic allocation
DFSOLSnn[1]	Online log data set (dual)	Archiving, dual logging
DFSTRAnn	External Trace	
DFSWADSn[1]	WADS data set and spares	Dual WADS logging, dynamic allocation
QBLKS[2]	Message queue blocks	Transaction traffic
SHMSG[2]	Short message	Message sizes
LGMSG[2]	Long message	Message sizes
IMSACBA/B	From staging IMS.ACBLIB	ACBGEN procedure execution, Online Change utility, use of DL/I address space
MSDBCP1[2, 3]	Fast Path MSDB checkpoint	Size of MSDBs
MSDBCP2[2, 3]	Fast Path MSDB checkpoint	Size of MSDBs
MSDBDUMP[2, 3]	Fast Path MSDB output	Size of MSDBs and operations
MSDBINIT[2, 3]	Fast Path MSDB input	MSDB Maintenance utility

Table 7. List of Online IMS Data Sets (continued)

ddname	Data Set/Content	Prerequisite
FORMATA/B[2]	From staging Format library	MFSUTL procedure execution, Online Change utility
IMSTFMTA/B[2]	Test message formats (PDS)	MFSTEST procedure execution, Online Change utility
IMSRDS	Restart data set	IMSCTF macro
MATRIXA/B	From staging security tables	Security Maintenance utility execution, Online Change utility
Comm Lines[2]	BTAM terminal addresses	LINEGRP, LINE macros
Databases[1, 3]	Database data sets	DATABASE macro, ACBLIB content, use of DL/I address space
IMSMON[1]	IMS Monitor output	IMSCTF macro
Spool names[2]	Spool output (IMS.SYSOnnn)	LINEGRP macro
PRINTDD	System output	Installation standards
DUMP	Diagnostic storage dump	Installation standards

Notes:

1. This ddname is not required if dynamic allocation macros are coded.
2. This ddname does not apply to DBCTL.
3. This ddname does not apply to DCCTL.

Initializing System Data Sets When Not Using Online Change

If you do not plan to use the online change function, you do not need to maintain the full set of staging, active, and inactive libraries. You only need to manage the staging libraries. You do not need to make copies for the active data sets—which would have exactly the same contents.

Specifying the IMS System Log

A series of DFSOLPnn DD statements specifies the IMS system log on which current activity for the IMS system is written. Each DD statement specifies an online log data set (OLDS). Instead of using DD statements, you can define some or all of the OLDS for dynamic allocation by using MVS and the IMS DFSMDA procedure. With this method, you have the option of setting aside spare OLDS to be dynamically allocated by the master terminal operator (using the `/START` command). Whether you use JCL statements or dynamic allocation, you must declare the list of numeric suffixes (nn), or OLDS identifiers, that are to be initially used by IMS. You do this with an OLDSDEF statement in the DFSVSMxx member of IMS.PROCLIB.

For write-ahead data sets (WADSs), you make preparations similar to those for OLDSs. You can use DD statements (DFSWADSn) or DFSMDA for dynamic allocation. With the latter, the master terminal operator can start and stop an individual WADS. For initial use by the IMS online system, you must define the WADS numeric suffixes in a WADSDEF statement in the DFSVSMxx member.

If you plan to use the additional integrity afforded to recovery operations when a duplicate system log is available, you need to include a second series of DD statements, DFSOLSnn, or define them with the DFSMDA macro. Also, specify MODE=DUAL on the OLDSDEF statement. For restart situations, dynamic backout, OLDS archiving, and BMP restart, IMS automatically switches to the secondary log if an I/O error is detected on the primary.

Related Reading: For more information on the system log strategy and its use in the IMS online system operation and recovery, see *IMS/ESA Operations Guide*.

Tuning the System Log Block Size

The online log data sets (OLDSs) corresponding to the DFSOLPnn and DFSOLSnn DD statements must be pre-allocated. Your installation chooses the value for the block size. The value must be a multiple of 2048 bytes. The specification uses fixed-blocked records. For the number of I/O buffers, you specify the BUFNO parameter on the OLDSDEF statement in the DFSVSMxx member of IMS.PROCLIB. The default number of buffers is 5. The OLDS block size must be the same for all data sets and must be specified in the pre-allocation step. (The DD statements that are generated for the IMS procedure that executes the control region use default DCB parameters.)

The OLDS block size should be relatively large and chosen to take advantage of the physical DASD device characteristics.

Related Reading: For more information on OLDS block size, see *IMS/ESA Installation Volume 1: Installation and Verification*.

If the active subsystem is being tracked by an RSR (Remote Site Recovery) tracking subsystem, the OLDS block size should not be larger than 32708 bytes. If the OLDS block size is larger, then log data is not sent to the tracking subsystem at the time that this data is written by the active subsystem. Instead, this log data is sent at a later time by the isolated log sender (ILS).

Related Reading: For more information on RSR processing, see “Chapter 12. Administering the Remote Site Recovery Complex” on page 347.

Message Queue Data Set Allocation

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The amount of direct access storage space allocated to the message queue data sets depends on how many transaction codes and logical terminal names are specified during system definition, and how many messages, both short and long, are to be held by the system during any period of time.

The amount of direct access storage space allocated to the message queue data sets can be changed prior to a cold-start of IMS. Reallocation of the message queue data sets with a warm-start requires the use of the FORMAT and BUILDQ parameters with either the `/NRESTART` or `/ERESTART` command. Allocating less space (than in the previous execution) prior to a `/NRESTART` or `/ERESTART BUILDQ` might cause the restart to abend.

You can allocate up to 10 data sets for the long message queue and 10 data sets for the short message queue. Each data set requires an additional DD statement.

Ensure that all data sets of a given message queue type are the same size. If the data sets have different sizes, the smallest size is used for all. This can reduce the available space of a message queue.

If you change the number of data sets, or if you rename any of the message queue data sets, you must restart the system.

The Queue Manager Concurrent I/O component provides multiple normal short and long message queue data sets. This facility is optional, and you can invoke it by providing 1 to 10 DD cards for the normal short and long message queue data sets.

The normal short and long message queues allow only one DD card for each.

In order to provide Queue Manager Concurrent I/O:

- IMS initialization allows multiple physical data sets to be viewed as one logical data set.
- You can view both the physical data set and the logical structure.

The Queue Manager Concurrent I/O component provides a performance enhancement by allowing the Queue Manager to perform parallel I/O on the message queues.

To prevent message queue overflow due to looping application programs, the Queue Manager and the Queue Space Notification exit routine (DFSQSPC0) monitor the number of buffers assigned to each unit of work (UOW). When a UOW exceeds its buffer limit, the Queue Space Notification exit routine takes action to prevent further inserts by that UOW, and an 'A7' status code is returned to the application program.

Related Reading:

- For more information on multiple message queue data sets, see *IMS/ESA Installation Volume 2: System Definition and Tailoring* or *IMS/ESA Installation Volume 1: Installation and Verification*.
- For more information on the 'A7' status code, see *IMS/ESA Application Programming: Database Manager*.

Additional Restrictions in an XRF Environment

Message queue data sets in an XRF environment have two additional restrictions:

- The number of data sets allocated for the short and long message queues must be the same on the primary and the alternate subsystems.
- The names for the message queue data sets must be different on the primary and alternate subsystems. These data sets cannot be shared between subsystems.

Message Queue Data Set Secondary Allocation

Several factors affect the usage of IMS.QBLKS records. For example, the requirement for multiple temporary destinations when using program isolation can cause an increase in the space requirements. The space requirements for the IMS.QBLKS data set depend on your installation.

The amount of direct access space required for the IMS.SHMSG and IMS.LGMSG data sets is dependent on message throughput. The disk space is reusable as soon as the message to which it was allocated has been processed and it is no longer required for recovery.

Message queue data set space should be allocated in terms of contiguous cylinders for most efficient operation. Secondary allocation is ignored unless the secondary space has been preallocated (that is, multiple volume data set with preallocated

space on both volumes). Allocate each message queue data set on a separate direct access device or next to each other, with IMS.QBLKS in the center, on the same direct access device.

Allocation for OSAM Data Sets

The recommended method of allocation for single or multiple volumes is through the use of JCL at the time the data set is loaded using the SPACE parameter.

If your installation control of DASD storage and volumes is such that the OSAM data sets must be reserved ahead of time, or if you decide that a message queue data set requires more than one volume, the OSAM data sets can be preallocated.

Pre-allocation has the following restrictions:

- DCB parameters should not be specified.
- If the data set is to be expanded beyond the preallocated space, a secondary quantity must be specified during pre-allocation. Queue data sets are constrained to only that space that is preallocated.

When a multiple-volume data set is preallocated, the method of allocation should allocate extents on all volumes to be used. The end of the data set must be correctly indicated in the DSCB on the last volume.

Related Reading: For more information on OSAM data sets, see *IMS/ESA Installation Volume 1: Installation and Verification*.

Message Queue Data Set Allocation Restrictions

If emergency restart procedures using BUILDQ are to be used, you must carefully reallocate logical record and data set spaces. The BUILDQ procedure always restores the message queue entries to the relative position in the respective queue data sets they had at the time they were saved. If the logical record or data set size has been decreased, situations exist in which it is impossible to perform the restart.

Related Reading: For more information on these situations, see *IMS/ESA Operations Guide*.

Restart Data Set Allocation

The restart data set is required. A minimum of 5 tracks must be allocated to the restart data set because it contains the checkpoint-ID table and other control information.

Defining Spooled SYSOUT Data Sets

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Message processing programs can require their own printed output to be separate from any system-wide SYSPRINT output. IMS provides for spooled output data sets, as well as a print utility (DFSUPRT0), which can be scheduled during execution of the online system.

You need to perform several definition steps to respond to the application and operational requirements, as described in the following sections.

Isolating the Spooled SYSOUT Requirements

You need to determine the LTERM names that the application programs will use. There might be output that is unique to a particular program or an agreement among several programs to use the LTERM for their online printed output.

You also need to obtain estimates of the volume of output so that you can make appropriate space allocations for the DASD data sets. Also, you need to assess the maximum output buffer size to handle the output without unnecessary buildup.

You should obtain some indication of the turnaround requirements for the output and whether the output is to be accumulated or produced in small batches. This information helps you decide how many data sets are needed to support the operation of each group of output (at least two data sets are recommended) and how to schedule the printing operation.

Allocating Required IMS.SYSnn Data Sets

You allocate 1 to 99 DASD data sets with appropriate space specifications. One of the factors that determines the availability of the data set for printing is the end-of-volume condition. You can either allocate sufficient primary space with any additional output planned to spill to alternative data sets, or you can allocate secondary space for unusual amounts of output.

Allocate and catalog these data sets along with your other online system data sets. The data set names are in the form IMS.SYSnn, beginning with IMS.SYS01.

Related Reading: For more information on IMS.SYSnn data sets, see *IMS/ESA Installation Volume 1: Installation and Verification*.

Defining Spool Line Groups in System Definition

In system definition, you specify a LINEGRP macro dedicated to SPOOL output. Associated with the LINEGRP macro are LINE, TERMINAL, and NAME macro specifications.

Related Reading: For more information on defining spool line groups, see *IMS/ESA Installation Volume 1: Installation and Verification*.

Initializing the RECON Data Set for DBRC

A necessary step in preparing for online operation is to ensure that the RECON data set, in which system log and database status is recorded, is ready. This part of installation should be coordinated with database administration personnel. The installation steps are:

1. Define two RECON data sets (named RECON1 and RECON2) and a spare data set (named RECON3).
2. Identify the database data sets to be tracked.
3. Record a starting set of data set levels.

Access Method Services parameters define the VSAM KSDS data sets referenced as RECON1, RECON2, and RECON3. Then the `INIT.RECON` command is used to write the required header record.

Related Reading: For more information on the allocation parameters, see *IMS/ESA DBRC Guide and Reference*. All three RECON data sets can be dynamically allocated.

Initializing Data Sets

A separate record is required for each data set of the databases that are to be controlled. The image copy, reorganization, and recovery operations call upon the current information in the RECON data set records. Using the `INIT.DBDS` command, you specify:

- The database name, the data set ddname, and the relevant data set names
- The number and reuse characteristics of image-copy data sets that are to be maintained
- Indicators of how the data sets are to be allocated by the IMS system
- The member names of procedures to execute database utilities

Using other commands in the INIT group creates other records for initial change accumulation, image copy, and system log data set names. INIT.DB is used to specify the share level for a database.

Related Reading: For more information on using the RECON data set for recovery, see *IMS/ESA Operations Guide*.

For system definition purposes you need to coordinate the entries on DATABASE macro statements, the online JCL requirements, and the database maintenance procedures.

DBRC RECON data sets from prior IMS releases must be upgraded to IMS/ESA Version 6 format with the RECON Upgrade utility.

Restriction: The DBRC skeletal JCL members supplied with IMS/ESA Version 6 are not downward compatible.

Related Reading: For information on the RECON Upgrade utility and the DBRC related commands, see *IMS/ESA DBRC Guide and Reference*.

Tailoring the IMS Procedure Library

This section addresses another part of the task of preparing the IMS online system for execution. The system requires not only control blocks built by the system definition process and suitably defined system data sets, but also a set of control parameters specified on the EXEC JCL statements of both the control region and each dependent region.

Part of the result of performing stage 1 of system definition is a set of updates to IMS.PROCLIB; these updates are listed in the stage 1 output. Stage 2 applies these updates to the library. You can tailor the contents of the members either before stage 2 or by direct maintenance against IMS.PROCLIB. Remember that you can set up default values for the parameters, but they can be re-specified at region startup with the use of the symbolic parameters. Initially, these parameters use values preset by system definition, but individual region control or tuning recommendations can override the initial or default values.

IMS.PROCLIB Members Generated by System Definition

The generated members of IMS.PROCLIB fall into several categories depending on their use. Generated IMS.PROCLIB members can be used for:

- Online operation
- Online initialization
- Online preparation and maintenance

- Batch operation
- Alternative batch execution
- Database definition and access
- Application program preparation

Related Reading: For information on specific PROCLIB members, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Controlling the IMS Procedure Library

Although many of the procedures generated in IMS.PROCLIB require alteration before they can be used in direct execution of the online system, they do provide a convenient start to the task of defining execution JCL. Many of the members have content that directly indicates the options you specified in the system definition stage 1 input. For example, the online execution member IMS includes a DD statement for IMSMON if the IMSCTF macro specifies LOG=MONITOR.

You should carefully examine the procedures generated as a result of your system definition.

Related Reading: For information on the differences from the listed procedures, and some explanatory notes about the JCL statements for several procedures that might apply to your generated library, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

This section describes the following information:

Assigning Procedure Names

You rename the IMS.PROCLIB members according to your installation's requirements. The names can follow a convention that suggests ownership by a particular application system or a convention that has an implied sequence. For example:

IMSIMAG	Initial procedure for image copies of database
IMSCTL	Control region startup (IMS renamed)
IMSTXLD	BMP to preload a transaction queue (IMSBATCH)
IMSMSG1	Message region startup (IMSMSG)
IMSBCH1	Low-priority BMP (IMSBATCH)
IMSMSG2	Second message region when required (IMSMSG)
IMSWT000	Spool output print procedure (named by IMS)

The procedure names are used by the system operator or master terminal operator to invoke the MVS job execution.

To develop these members, you need to either rename the members in IMS.PROCLIB or create new members. In some installations, the procedures are added to SYS1.PROCLIB. One option of the NODE keyword on the IMSGEN macro allows you to substitute an alternative library naming convention, so that your base procedure library can be named LEGAL.PROCLIB.

Initializing Your Procedure Library

Given your requirements for the members of the procedure library, you now need to adjust the exact JCL content. The updates you apply follow naming conventions for your installation as well as the required DD statements. In all procedures, you can add JCL comment statements for additional documentation.

Related Reading: For more information on the tailoring actions of each PROCLIB member, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Preparing for IMS Job Execution

In preparation for executing the IMS procedure as a system task, the members IMS, IMSRDR, DBC, DCC, DBRC, and DLISAS are moved to SYS1.PROCLIB. Additional generated members of IMS.PROCLIB, especially IMSMSG and the set of IMSWTnnn members for spool output, need to be tailored to satisfy your installation's requirements. IMSMSG invokes the procedure DFSMPR, and the IMSWTnnn members invoke DFSWTnnn. To enable message region and spool output jobs to be started with IMS commands or from the system console, these members, after tailoring, are moved to the IMS.JOBS data set,which is concatenated with SYS1.PROCLIB.

Preparing PROCLIB Member DFSPBxxx

PROCLIB member DFSPBxxx contains IMS control region execution parameters. The values specified on the EXEC statement override any parameters specified on DFSPBxxx.

To build a DFSPBxxx member, use the sample member that generation creates. The sample members are DFSPBIMS for IMS DB/DC, DFSPBDBC for DBCTL, and DFSPBDCC for DCCTL. The sample members contain all the valid parameters for the specified IMS control region.

To use the DFSPBxxx member, code RGSUF=xxx on the invocation of the IMS procedure.

Related Reading: For more information on the DFSPBxxx member and the IMS procedure, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Creating PROCLIB Member DFSFDRxx

PROCLIB member DFSFDRxx specifies the options that Fast DB Recovery uses. You can specify multiple instances of DFSFDRxx in IMS.PROCLIB, but each DFSFDRxx member must have a unique, two-digit suffix. Fast DB Recovery, IMS, or DBC procedures identify which DFSFDRxx member to use.

Related Reading: For more information on the DFSFDRxx member and Fast DB Recovery, IMS, and DBC procedures, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Tailoring Fast Path Execution Procedures in DB/DC

or DCCTL

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

As a result of defining Fast Path application programs in system definition macros, the contents of IMS.PROCLIB include two procedures for executing Fast Path dependent regions in a DB/DC or DCCTL environment:

IMSFP To execute a region containing a Fast Path application program

FPUTIL To execute online DEDB utilities

You must tailor both of these procedures to the requirements of individual programs. Also, several of the other members of the procedure library have EXEC statement parameters that specifically apply to Fast Path or need to reflect Fast Path requirements. The additional tailoring actions are summarized in Table 8.

Table 8. Tailoring Actions for Fast Path Procedures

PROCLIB Member	Tailoring Action
IMS	Specify buffering and output thread limits Re-specify region size Add appropriate DEDB DD statements Allocate MSDB initialization data set
DFSMPR	Specify buffering reserved for this region
IMSBATCH	Specify buffering reserved for this region
IMSFP	Specify the application program and region size Specify buffering reserved for this region
FPUTIL	Specify the DEDB name and restart indicator Provide the utility control statements
DFSFIXxx	Page fix list for Fast Path control blocks
DBFMSDBx	List of MSDBs and segments to be loaded

Related Reading: For more information on each of these PROCLIB members, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Tailoring Fast Path Execution Procedures in DBCTL

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

After defining Fast Path application programs in system definition macros, the contents of IMS.PROCLIB in a DBCTL environment include FPUTIL, a procedure that executes online DEDB utilities.

This procedure must be tailored to the requirements of individual programs. Also, several of the other members of the procedure library have EXEC statement parameters that specifically apply to Fast Path or need to reflect Fast Path requirements. The additional tailoring actions are summarized in Table 9.

Table 9. Tailoring Actions for Fast Path Procedures in a DBCTL Environment

PROCLIB Member	Tailoring Action
DBC	Specify buffering and output thread limits Re-specify region size Add appropriate DEDB DD statements
IMSBATCH	Specify buffering reserved for this region

Table 9. Tailoring Actions for Fast Path Procedures in a DBCTL Environment (continued)

PROCLIB Member	Tailoring Action
FPUTIL	Specify the DEDB name and restart indicator Provide the utility control statements
DFSFIXxx	Page fix list for Fast Path control blocks

Related Reading: For more information on each of these PROCLIB members, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Controlling Procedure Library Modifications

Using operator commands keeps the initial operating instructions simple and avoids complex symbolic parameter data entry. However, you must control the exact content of the JCL residing in IMS.PROCLIB. For example, the master terminal operator enters:

```
/START REGION IMSBCH1
```

The procedure IMSBCH1 must be correctly coordinated to a known BMP, appropriate PSB and transaction queue, other system options, and identifying parameters.

Your control responsibilities can include auxiliary procedures for database reorganization, recovery, or system output control. Further, the modification level of each procedure must be coordinated to the actual production environment. For example, if an application program is modified and requires more dependent region storage, you must coordinate the program library and IMS.PROCLIB changes. One technique is to include JCL comment statements that document the date and the reason for the change.

Take special care to check the physical changes you make in procedure library members. Many of the DD statements extend over several input records and involve positional parameters, so you need to make more than a cursory examination of a change. You can use a data dictionary to record and maintain the procedure library members. Changes can then be checked at the terminal or by reviewing listings of the changed members.

Specifying EXEC Statement Parameters

The EXEC statement parameter categories are:

- Database and PSBs
- Data communications
- System control and performance
- Recovery and restart
- Security options

The EXEC statement parameters are presented in the following sections, within these categories shown. These categories are described for the control region, for the message processing region, and for the batch message region.

- “Control Region Parameters” on page 95
- “Message Processing Region Parameters” on page 103
- “Batch Message Processing Region Parameters” on page 105
- “Fast Path Dependent Region Parameters in DCCTL or DB/DC” on page 108
- “Fast Path Parameters in BMP and CCTL Regions in DBCTL” on page 109

- "Online DEDB Utility Region Parameters in DCCTL or DB/DC" on page 110
- "Online DEDB Utility Region Parameters in DBCTL" on page 110

Related Reading: For more information on these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Control Region Parameters

Use the PARM1= and PARM2= parameters on the EXEC statements for the control region to specify the JCL for IMS online execution. For optimal performance, it is recommended that you use PARM1= and PARM2= parameters rather than creating your own JCL. These parameters are specified symbolically for the control region.

EXEC Parameters for Database Buffers

The VSPEC parameter enables you to point to member DFSVSMxx in IMS.PROCLIB; this member predefines the buffer pool requirements for databases that use OSAM or VSAM as the access method.

Related Reading: For more information on the specification for the subpool sizes, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

EXEC Parameters for DMB and PSB Buffers

Several parameters enable you to override the size of buffer pools to hold DMBs and PSBs that were predefined during system definition. Performance analysis often results in a request to increase DMB or PSB buffer space. To change DMB or PSB buffer space, you can override the system definition values.

Related Reading: For more information on overriding these values, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Fast Path EXEC Parameters in DCCTL or DB/DC

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

The PARM1= and PARM2= positional parameters for the control region's EXEC statement can also be used to specify the Fast Path parameters shown in Table 10.

The parameters determine:

- MSDB load requirements
- Overrides of buffer sizes
- DEDB options

Table 10. Categories and Purpose of Fast Path Control Region Parameters

Category	Parameter	Purpose
MSDB load	MSDB	Specify the DBFMSDBx suffix
Database buffer sizes	BSIZ	Specify the common size of a buffer
	DBBF	Specify the maximum number of buffers
	DBFX	Specify system buffer allocation

Table 10. Categories and Purpose of Fast Path Control Region Parameters (continued)

Category	Parameter	Purpose
DEDB options	OTHR	Specify the number of DEDB updates that can be concurrently waiting for I/O after sync point
	LGNR	Specify the maximum DEDB buffer alterations before CI logging
EPCB	EPCB	Specify the size of the EPCB pool
EMHL	EMHL	Specify the size of the EMHL buffer

When an emergency restart is performed, the values specified for the EXEC parameters MSDB, BSIZ, DBBF, OTHR, and LGNR, and the contents of the member DBFMSDBx, must remain unchanged from the last normal start. (These values are used when reestablishing buffer contents from checkpoint records.)

MSDB Loading: The MSDB parameter enables you to control which MSDBs must be loaded for the current online session and how many segments each MSDB requires. The parameter is generated with a null value, implying that the MSDBs are loaded directly from a Fast Path system data set. You specify a 1-character suffix that points to an IMS.PROCLIB member, DBFMSDBx, containing the detailed requirements.

If the MSDB requirements for an online session are a subset of the MSDBs, or if the number of segments for any MSDB is to be increased to reserve space for dynamic terminal-related MSDBs, you must coordinate the content of the DBFMSDBx member and the MSDB parameter value. The control statements in the DBFMSDBx member must explicitly name all MSDBs required to be loaded. The control statements also enable you to individually select MSDBs for page fixing.

LU 6.2 devices allow read-only access to dynamic MSDBs. LU 6.2 devices cannot access terminal-related MSDBs.

Terminals defined dynamically with ETO are not available to terminal-related MSDBs.

Database Buffers: Using the DBBF and BSIZ parameters, you declare the total buffers available for the IMS online system's processing of MSDB and DEDB activity. Dependent regions then claim portions of this allocation. The parameters have the following characteristics:

DBBF
Specifies the maximum number of these buffers, from 1 to 9999. The storage is obtained as needed from the extended common storage area (ECSA). You might need to adjust the ECSA or region due to your requirements.

Your DBBF value adjusts the second subparameter value specified for the BFALLOC keyword in the system definition FPCTRL macro. A null value is generated for the IMS procedure, so that the number of buffers is set to the default FPCTRL value. If you override, you must adjust for any change in DBFX specification.

BSIZ
Specifies buffer size in bytes. Choose a value corresponding to the largest DEDB control interval size. BSIZ is generated as a null value to invoke the

system definition default value. The default value is usually sufficient, because your largest DEDB CI size is likely to be unchanged.

DBFX
Reserves a subset of the total Fast Path buffers for general system use. The parameter specifies the number of buffers to be page fixed when the first Fast Path dependent region is started. A null value is generated to invoke the system definition default, specified as the first parameter value for the BFALLOC keyword on the FPCTRL macro.

DEDB Options: The OTHR parameter specifies the number of asynchronous output threads. You can revise the estimate of DEDB updates that occupies buffer space while they are waiting to be committed to the area data sets after sync point logging. Your revision overrides the value given for the OTHREAD keyword on the FPCTRL macro, and can range from 1 to 255 but cannot be more than the specified value of the MAXPST parameter on the IMS procedure.

The LGNR parameter determines a maximum number of individually logged DEDB alterations made in a buffer. Above that number, the whole control interval (CI) is logged. The default value is 7, and you can specify a number from 7 to 99. If the application program is using large CIs, the LGNR value can be increased to economize on the system log activity.

Fast Path EXEC Parameters in DBCTL

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

The PARM1= and PARM2= positional parameters for the control region's EXEC statement that are additionally specified for Fast Path are shown in Table 11.

The parameters determine:

- Overrides of buffer sizes
- DEDB options

Table 11. Categories and Purpose of Fast Path Control Region Parameters

Category	Parameter	Purpose
Database buffer sizes	BSIZ	Specify the common size of a buffer
	DBBF	Specify the maximum number of buffers
	DBFX	Specify system buffer allocation
DEDB options	OTHR	Specify the number of DEDB updates that can be concurrently waiting for I/O after sync point
	LGNR	Specify the maximum DEDB buffer alterations before CI logging
EPCB	EPCB	Specify the size of the EPCB pool

When an emergency restart is performed, the values specified for the EXEC parameters BSIZ, DBBF, OTHR, and LGNR must remain unchanged from the last normal start. (These values are used when reestablishing buffer contents from checkpoint records.)

Database Buffers: Using the DBBF and BSIZ parameters, you declare the total buffers available for the IMS online system's processing of DEDB activity. Dependent regions then claim portions of this allocation. The parameters have the following characteristics:

DBBF
Specifies the maximum number of these buffers, from 1 to 9999. The storage is obtained as needed from the extended common storage area (ECSA). You might have to adjust the ECSA or region due to your requirements.

Your DBBF value can adjust the second subparameter value specified for the BFALLOC keyword in the system definition FPCTRL macro. A null value is generated for the IMS procedure, so that the number of buffers is set to the default FPCTRL value. If you override, remember to adjust for any change in DBFX specification.

BSIZ
Specifies the buffer size in bytes. Choose a value corresponding to the largest DEDB control interval size. BSIZ is generated as a null value to invoke the system definition default value. The default value is usually sufficient, because your largest DEDB CI size is likely to be unchanged.

DBFX
Reserves a subset of the total Fast Path buffers for general system use. The parameter specifies the number of buffers to be page fixed when the first Fast Path dependent region is started. A null value is generated to invoke the system definition default, specified as the first parameter value for the BFALLOC keyword on the FPCTRL macro.

DEDB Options: The OTHR parameter specifies the number of asynchronous output threads. It allows you to revise the estimate of DEDB updates occupying buffer space while waiting to be committed to the area data sets after sync point logging. Your revision overrides the value given for the OTHREAD keyword on the FPCTRL macro, and can range from 1 to 255 but cannot be more than the specified value of the MAXPST parameter on the DBC procedure.

The LGNR parameter determines a maximum for the number of DEDB alterations made in a buffer that are individually logged. Above that number, the whole control interval (CI) is logged. The default value is 7, and you can specify a number from 7 to 99. If the application program is using large CIs, the LGNR value can be increased to economize on the system log activity.

Other Database Performance Options

Use the system definition to specify individual database DMBs and PSBs to be resident. You can override this specification with the parameter RES. When you execute a test system or if storage is temporarily constrained, it might be useful to specify RES=N so that all the control blocks are not made resident.

Data Communications EXEC Parameters

The set of parameters in the IMS procedure that are related to data communications are:

- Overrides of predefined parameters
- ETO parameters
- Performance options

Overrides of Predefined Parameters (FBP, QBUF, SAV, EMHL, RECASZ, RECA)
Monitoring communication traffic often results in a decision to increase

buffer space or the number of buffers. The following execution parameters are provided to override the size and number of some of the buffer pools predefined at system definition:

FBP
Overrides the size specified on the FORMAT(size1) keyword of the BUFPOOLS macro.

QBUF
Message queue buffer. Adjusts the size of the message queue buffer pool. QBUF enables you to override the number of blocks to hold messages in the control program's storage. By increasing the number of buffers used for the message queue data sets, you can reduce the frequency of I/Os. The default number of buffers is specified in the MSGQUEUE macro.

EMHL
Specifies the EMH buffer length for Fast Path transactions.

RECASZ
Specifies the size of the Receive Any buffer.

RECA
Overrides the number of Receive Any buffers predefined in system definition by the RECANY keyword of the COMM macro.

SAV
Specifies the allowed maximum number of concurrently active device I/Os.

ETO Parameters (ALOT, ASOT, CHTS, LHTS, NHTS, UHTS, DLQT, ETO, DSCT)
Use the following parameters to define ETO options:

ALOT
Specifies the amount of time allotted before a terminal in session is automatically logged off if no user successfully signs on.

ASOT
Specifies the amount of time allotted before a signed-on user is automatically signed off if no input or output activity occurs.

CHTS
Specifies the number of slots for the CCB hash table.

LHTS
Specifies the number of slots for the CNT/LNB/RCNT hash table.

NHTS
Specifies the number of slots for the VTCB hash table.

UHTS
Specifies the number of slots for the SPQB hash table.

DLQT
Specifies the number of days allotted before a queue containing data that has not been allocated is classified as a potential dead-letter queue.

ETO
Specifies whether terminals and queues that are not defined to IMS are to be supported.

DSCT
Specifies the suffix of the descriptor member DFSDSCTy.

Performance Options (VAUT, NLXB, FESTIM)

Use the following parameters to define performance options:

VAUT
Specifies the use of VTAM-Authorized Path.

NLXB
Specifies that parallel sessions are added during system startup.

FESTIM
Overrides the timeout value for Front End Switching; this value is predefined during system definition.

System Control and Performance EXEC Parameters

The second parameter in the control region procedure, RGSUF, becomes a 3-character suffix for a DFSPBxxx member in IMS.PROCLIB. You can define all online parameter default values in that member rather than individually setting symbolic parameter values on the EXEC statement in the control region procedure.

Related Reading: For more information on control region parameters in the procedures and in DFSPBxxx members, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The remaining positional parameters affect control of the system resources as follows:

Identification of nucleus (SUF)

You must specify the SUF control region parameter to specify which IMS configuration is to be executed. The default value is 0 (zero).

Number of active regions (PST)

Use the PST parameter to override the expected number of regions that are in operation during the online execution. Additional regions can be dynamically allocated, up to the maximum allowable number permitted by your operating system.

Performance-related options (FIX, PRLD, EXVR)

Three parameters contribute to general performance strategy. The FIX parameter allows you to point to themembers DFSFIXxx and DFSDRFxx in IMS.PROCLIB. All modules and control blocks that are to be page fixed and put into DREF strings are described with these members. Similarly, the PRLD parameter points to member DFSMPLxx,where a list of all preloaded modules is given. The EXVR parameter allows you to page fix buffers used for the management of message queues.

MVS dependent options (SRCH, LSO=Y)

Several parameters apply only to the MVS operating system. The SRCH parameter allows you to take advantage of any special library structure to optimize the search for loaded modules. You can override the default with a value of 1 if you want the JPA and LPA to be searched before IMS program libraries.

To reduce the amount of CSA storage available to IMS running under MVS, override the null value generated for the LSO parameter by specifying LSO=Y. This causes some IMS modules used for DL/I processing and some control blocks to be loaded into the control program's private storage.

Inclusion of the DL/I address space (LSO=S, SOD)

A further variation of the local storage option is to use the DL/I separate address space. You do this by specifying LSO=S. This address space

contains most of the DL/I code, control blocks, and database buffers for full-function databases. MVS cross memory services are used.

In the area of operations for MVS, you can specify the output class of a spin-off dump using the SOD parameter. You must override the null value generated for the IMS procedure. This should be standard practice for MVS installations, because it allows the analysis in hard-copy form of the status of real storage immediately after abnormal termination of the control region or dependent regions.

Abnormal termination output (FMTO)

In an online environment, you can request the following types of dumps for errors that terminate IMS: SDUMP, SYSMDUMP, SYSABEND, or SYSUDUMP. You do this by using the FMTO startup parameter in combination with MVS dump DD statements.

You can also choose whether dumps are produced for some errors that do not terminate IMS. This choice depends on the type of failure as well as the FMTO parameter option and IMS spin-off and MVS dump DD statements that are selected.

Related Reading: For more information on the FMTO parameter and on using dumps in these situations, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

For SYSMDUMP, you must establish operational procedures for saving and formatting dumps because of the risk of overlaying a SYSMDUMP if IMS is restarted before the previous SYSMDUMP is transferred.

Inclusion of an Internal Resource Lock Manager (IRLM, IRLMNM, UHASH)

If an IMS online system uses IRLM as the lock manager or participates in block-level sharing, you request the use of an IRLM subsystem by coding IRLM=YES.

Related Reading:

- For more information on IRLM requirements, see “Tailoring Execution JCL” on page 318.
- For the naming convention for IRLM, see “Identifying the IRLM” on page 334.

If your online system includes Fast Path, a related parameter, UHASH, is used in connection with the IRLM lock manager. This parameter specifies the name of an alternative hashing module. If you did not use the default name of DBFLHSH0 in the system definition, the appropriate name must be specified. This name cannot be changed across a restart. If multiple IMS subsystems are taking part in data sharing and are using this option, the module name must be identical in all systems.

Identification of external subsystems that can be attached (SSM)

The SSM parameter is used to reference a member in IMS.PROCLIB; the member identifies the external subsystems (DB2, for example) that can be accessed from application programs executing in dependent regions. Specify a suffix that, together with the currently assigned name for IMSID, forms the member name. The member contains entries, each identifying an external subsystem—its MVS subsystem name. All external subsystems that might be accessed from programs executing in dependent regions must have corresponding entries.

The SSM parameter for each dependent region need not be coded to work with the IMS control region member. identified to the IMS control region,

specify the name of a null member (no entries). To allow access to one or more particular external subsystems, the dependent region SSM parameter needs to point to a member containing only those specific (and matching) entries.

Modifying Storage Pool Definitions for the Storage Manager

You can set an upper expansion limit for the HIOP, CIOP, SPAP, LUMP, LUMC, FPWP, and EMHB storage manager pools by using the appropriate execution parameters. IMS establishes these storage pool definitions without an upper expansion limit, because they are dynamic storage pools that expand and contract as needed during execution.

Use caution in specifying upper expansion limits. If an upper limit is too low, IMS might abend. Under normal circumstances, a pool should never reach its upper limit. The intent of the upper limit is to keep pools from consuming so much storage that an out-of-storage condition occurs.

Use the SPM=nn parameter to specify the suffix for DFSPMnn. DFSPMnn identifies the IMS.PROCLIB member that overrides the storage pool definitions established by IMS.

Related Reading: For more information on specifying the individual entries and constructing the IMS.PROCLIB member, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Recovery-Related EXEC Parameters

The WADS parameter indicates whether a single write-ahead data set (WADS) is to be used by the IMS online system, or whether dual data sets are to be used. Using dual data sets helps protect the integrity of the online logging.

Use the ARC parameter to specify whether automatic archiving of the online log data sets (OLDS) is to be performed. Automatic archiving is recommended, although your installation can arrange for the MTO to monitor the availability of OLDS and perform archiving when necessary. You need to coordinate this parameter with your installation's operating procedures.

The DBRCNM parameter is used to specify an alternative name for the DBRC region start procedure.

Use the AUTO parameter to specify automatic restart for the IMS online system.

Use the QTU and QTL parameters to adjust the upper and lower limits in the Queue Space Notification exit routine (DFSQSPC0).

Related Reading: For more information on the IBM-supplied Queue Space Notification exit routine, see *IMS/ESA Customization Guide*.

Security-Related EXEC Parameters

The EXEC parameters for the control region include a way to control the kind of security checking that is done during the current execution. The parameters act as switches for different types of security. They also determine what flexibility the MTO has to override the choice of security checking. You must coordinate the setting of these switches with both overall security design and operational procedures. The parameters are TRN, SGN, RCF, and ISIS.

The default values generated for the IMS procedure all specify no security. You need to reset them to match the security design your installation requires. The parameter values and their meanings are given in “Controlling System Startup” on page 143.

You also need to coordinate the level of the security tables with the suffix identifier for the nucleus. Additional operational restrictions for the master terminal operator are explained in “Security Considerations for the Master Terminal” on page 124.

The IMSID parameter is related to both security and operations. The parameter value uniquely identifies the control region. Dependent regions that are to execute under control of this nucleus must specify the same identifier. The generation default is IMSA.

The name chosen should be unique to other subsystems executing in the MVS system, including any batch executions, and they should not be the same as the procedure name. This is recommended because any message to the console originating from subsystem execution is identified by that name. This uniqueness helps avoid confusion as to the source of the message.

Message Processing Region Parameters

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Use the PARM= positional parameters for the message processing region’s EXEC statement to control:

- Database and PSB
- Data communication
- Region control and performance
- Recovery and restart
- Security options

Related Reading: For more information on these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

PSB-Related EXEC Parameter

Use the PCB parameter to specify the size of the buffer used by inter-region communication to hold a copy of the user’s PCBs. The IMSMSG procedure assumes that the default buffer size, an area equal to the largest PCB contained in any PSB in the active IMS.ACBLIBA/B library, is to be allocated.

The VALCK parameter default signifies that address validity checking is not to be performed for DL/I calls issued by the application programs in this region. (An address is invalid if it is either lower than the lowest address not in the MVS nucleus or higher than the highest address in virtual storage.) With adequate testing of, and controls over, the DL/I call parameter coding, address validity checking should not be necessary.

Data Communication EXEC Parameter

The set of 4 parameters, CL1, CL2, CL3, CL4 is required for a message region. You specify 4 transaction classes, each expressed as a 3-digit number. The first message class for the region causes all messages assigned to that class to be selected first as eligible for scheduling. Only when all possibility of scheduling a transaction in that class has been exhausted does scheduling begin for the second

message class. Priorities determine the order that programs are to be selected for scheduling into the region. The message classes you specify need to be coordinated with your transaction scheduling rules and the numbers entered with the PRTY keyword on the TRANSACT macro.

Region Control EXEC Parameters

The first positional parameter is coded MSG for a message processing region. Programs are scheduled in these regions automatically when transactions are encountered on the queue, if the message class priority is suitable.

The parameter OPT helps you control the region startup. If the control region is terminating or not active when the MPP region is invoked, you can have the master terminal operator decide whether to start the MPP region again (the default), let it wait until the control region is ready, or cancel it. With good operator control, the operator's response should be adequate.

Use the STIMER parameter to invoke timer facilities. The default is to record processor time for the duration of program execution, including DL/I processing time. Because a timeout of a message region causes an abnormal termination of the message region, a better strategy is to have processing limits coded within the application program.

Use the SSM parameter to point to a member in IMS.PROCLIB that identifies the external subsystems (for example, DB2) that can be accessed from this MPP region. To prevent any access to such a subsystem from an MPP scheduled into this region, use the name of a null member (a member having no entries). The default, a null value, allows the MPP region to attach to any of the subsystems declared to the IMS control region. If necessary, coordinate this parameter with the corresponding SSM parameter in the IMS procedure. Refer to the description in section "System Control and Performance EXEC Parameters" on page 100.

Use the APARM parameter to specify execution time parameters that are unique to this dependent region. This parameter specifies a character string for the application program or Data Capture exit routine.

To print out the storage dump immediately after the abend of a message region, specify the correct output class for the SOD parameter.

Performance-Related EXEC Parameters

Performance-related parameters for a message region include OVLA, DBLDL, PRLD, PREINIT, VSFX, VFREE, and PWFI. You can choose options with these parameters to improve system performance.

Related Reading:

- For more information on each of these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For more information on improving your system's performance, see "Chapter 8. Tuning Your System" on page 265.

Recovery-Related EXEC Parameters

The TLIM parameter addresses a problem with an application program that causes an abnormal termination. Because the program might be scheduled many times into a region due to transactions in the queue, you need to be able to stop the operation of this region. The value for TLIM is the maximum number of abnormal terminations permitted.

In the case where the application program has a SPIE in effect, the SPIE option allows it to remain on during the DL/I call or for it to be turned off during the DL/I call and reinstated when returning to the application program. For performance reasons, it is undesirable to turn the SPIE option on and off for each DL/I call. With PL/I Release 5, you can use the PL/I SPIE facility without having IMS reset the SPIE on each DL/I call.

Security-Related EXEC Parameters

The AGN parameter allows you to specify the application group name (AGN); the default is no AGN. The application group name is associated with a set of transactions, PSBs, or LTERMs that are authorized to be used by this region. The authorization is made by declaring these resources and the AGN in the Security Maintenance utility input. You include the same name to invoke IMS resource access security for this region. The DFSMPR procedure generates this as a null parameter. If other regions are being controlled through the use of AGNs, and if you omit this parameter, you risk permitting unauthorized access by programs executing in this region.

The parameter IMSID is related to security and operations. You specify the identifier for the name of the control region (the name given for the IMSID parameter). If the value does not match any current IMSID of an operating control region, the message region is not scheduled.

Batch Message Processing Region Parameters

Use the PARM= positional parameters for the BMP region's EXEC statement to control:

- Database and PSB
- Data communications
- Region control and performance
- Recovery and restart
- Security options

PSB-Related EXEC Parameters

The MBR parameter is required. It specifies the name of the program and is often the same as the PSB name. The BMP has flexibility in using a program and PSB combination. This combination allows you to test modifications of the BMP using a temporary program name. You can also use a different PSB with the same program.

The PSB parameter is optional if it matches the MBR name. an APPLCTN macro has been included for the PSB specifying BATCH as the program type. You should identify application programs that have the ability to These programs often require larger amounts of virtual storage and you might need to adjust the size of the region.

The TEST parameter is required, but the IMSBATCH procedure generates a 0 (zero) to specify that validity checking of the addresses in a DL/I call is not performed. (An address is invalid if it is either lower than the lowest address not in the MVS nucleus or higher than the highest address in virtual storage.) Adequate testing of the program, and controls over the DL/I call parameter list coding, should make the generated option—no validity checking—acceptable.

Data Communication EXEC Parameters

For the batch message program, you have the flexibility of declaring that the input transaction queue is to be made available to the program at execution time. You do this by specifying one transaction code as the value for the IN parameter.

Some BMP programs do not access a message queue but do have a requirement to send output to a terminal or to generate transactions to be processed by other application programs. You specify the LTERM name or transaction code, as appropriate, on the OUT parameter. accessing message queues. When you specify the transaction code for the IN parameter, the program has no restrictions on generated transactions or output messages.

Region Control EXEC Parameters

The first positional parameter is coded BMP for a batch message processing region. These regions are not scheduled automatically, but must be invoked by the operator.

The OPT parameter is required. It helps you control the start of a batch message region. If the control region is not active when the BMP region is invoked, you can decide to let it wait, cancel it, or ask the operator to make the decision. This could occur when JCL (in the MVS job stream for the BMP region) is scheduled before the control region has completed its initialization or is terminating. There is a risk in specifying ‘wait’ because the MVS resource is reserved until the control region resumes. If you are starting the region from the master terminal, the default generated for the IMSBATCH procedure is satisfactory.

Use the PRLD= parameter to specify the suffix for the IMS.PROCLIB member, DPSMPLxx, which lists the preloaded modules. No performance improvement is provided for BMP regions that do not have other programs scheduled in them.

Use the PREINIT parameter to specify the suffix of the IMS.PROCLIB member, DFSINTxx.

The STIMER and CPUTIME parameters are optional; they are generated as null parameters, and the default is no use of the timer. The parameters are used together. If you want to limit the duration of processor time the batch message program can run, you specify STIMER=1 and give CPUTIME a value equal to a number of minutes. The measured time includes any DL/I processing performed in the region. You should use this technique to limit the batch message execution in preference to a time limit on the job card. This is because an abnormal termination in the dependent region, caused by excessive processor time, can also terminate the control region. Excessive processor time causes a user abnormal termination (U0240) for only the dependent region.

The PARDLI parameter is optional; the IMSBATCH procedure generates it as a null parameter. The default value causes DL/I processing to be executed in the dependent region. You might want to specify that the control region does all DL/I processing for the program (PARDLI=1). This gives the program DL/I service at a higher priority, although other DL/I service times could be adversely affected. Setting the PARDLI parameter also affects the BMP region and timing out (system X22 abends). By specifying PARDLI=1, you can prevent a corresponding control region system 113 abend.

The DIRCA parameter is required. It specifies the number of blocks of 1024 bytes that must be reserved for inter-region communication. You base the size calculation on the number of bytes required to hold the largest PCB contained in the PSB. The

IMSBATCH procedure generates a default value of 000, which causes an area equal to the largest PCB in any PSB in the active IMS.ACBLIBA/B library to be used. You might prefer to adjust the size downward.

Use the SSM parameter to reference a member in IMS.PROCLIB that identifies the external subsystems (for example, DB2) that can be accessed from this BMP region. To prevent any access to such a subsystem from the BMP, use the name of a null member (a member having no entries). The default, a null value, allows the BMP region to attach to any of the subsystems declared to the IMS control region. If necessary, coordinate this parameter with the corresponding SSM parameter in the IMS procedure. Refer to the description in “System Control and Performance EXEC Parameters” on page 100.

The FMTO parameter and the type of the MVS dump DD statements selected determine whether IMS dumps are formatted online or offline.

Use the APARM parameter to specify execution time parameters that are unique to this dependent region. This parameter specifies a character string for the application program or Data Capture exit routine.

Related Reading: For more information on the region control EXEC parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Recovery-Related EXEC Parameters

For a BMP program, you can use the CKPTID parameter as a restart position for the program processing. The IMSBATCH procedure generates CKPTID as a null parameter. To invoke the restart, you need to create a special version of the procedure containing the exact checkpoint identification.

Related Reading: For more information on invoking the restart, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

In any case, the restart design must allow for restart and such things as MVS file positioning that you do not control.

Related Reading: For more information on restart for BMPs that use extended checkpoint, see *IMS/ESA Operations Guide*.

If the application program has a SPIE in effect, the SPIE option allows it to remain on during the DL/I call, or to be turned off during the DL/I call and reinstated when returning to the application program. For performance reasons, it is undesirable to turn the SPIE option on and off for each DL/I call. With PL/I Release 5, you can use the PL/I SPIE facility without having IMS reset the SPIE on each DL/I call.

Security-Related EXEC Parameters

The AGN parameter allows you to specify the application group name (AGN); the default is no AGN. The application group name is associated with a set of transactions, PSBs, or LTERMs that are authorized to be used by this region. The authorization is made by declaring these resources and the AGN in the Security Maintenance utility input. You include the same name to invoke IMS resource access security for this region. The IMSBATCH procedure generates this as a null parameter. If other regions are being controlled through the use of application group names, you risk permitting unauthorized access by this batch message program if you omit the AGN parameter.

The parameter IMSID is related to both security and operations. If this parameter does not match the current IMSID of any operating control region, the batch message region is not scheduled.

Fast Path Dependent Region Parameters in DCCTL or DB/DC

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

The PARM= positional parameters for Fast Path dependent region's EXEC statements are shown in Table 12.

Table 12. Categories and Purpose of Fast Path Dependent Region Parameters

Category	Parameter
Database and PSB	NBA OBA MBR PSB
Data communications	none
Region control and performance	IFP OPT PRLD PREINIT STIMER DIRCA CPUTIME DBLDL SSM ALTID PARDLI
Recovery and restart	TLIM SOD
Security options	AGN IMSID

The EXEC statement's first positional parameter, IFP, causes this region to be active for Fast Path processing only. The MBR parameter specifies the name of the message-driven application program.

Two database-related parameters, NBA and OBA, control how many of the total Fast Path buffers this region can appropriate for its use.

The NBA parameter specifies how many buffers are reserved for DEDB and MSDB processing by this region. This normal allotment is obtained at region startup and must be available from the total number specified for the control region. Although you can specify a number from 1 to 999, you coordinate the value across all regions that are to be concurrently active, so that the total can be specified by the DBBF parameter for the control region. If too few buffers are available from the DBBF total, the dependent region abends.

The OBA parameter specifies how many buffers are requested from the control region as overflow when the normal allotment is used up. The control region page

fixes the largest number of overflow buffers specified for all active dependent regions and makes these available to only one program at a time.

Both the NBA and OBA parameters are generated with a 0 (zero) value so that you must re-specify the individual values.

Related Reading:

- For more information on DEDB buffer considerations in a DBCTL environment, see *IMS/ESA Administration Guide: Database Manager*.
- For more information on Fast Path dependent region parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Fast Path Parameters in BMP and CCTL Regions in DBCTL

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

The PARM1= and PARM2= positional parameters for a BMP region's EXEC statements are shown in Table 13. Also shown are the Fast Path parameters in the DRA startup table that the CCTL must use when it connects to a DBCTL environment.

Table 13. Fast Path Dependent Region Parameters for DBCTL

Category		Parameter	Purpose
BMP	Database and PSB	NBA	Specifies the number of database buffers
		OBA	Specifies the number of overflow buffers
		MBR	Specifies name of message-driven program
		PSB	Specifies PSB name
CCTL	Database	CNBA	Total number of buffers for this CCTL
		FPB	Number of database buffers each thread uses (from the CNBA total)
		FPOB	Number of overflow buffers each thread might need

Two database-related parameters, NBA and OBA, control how many of the total Fast Path buffers this region can appropriate for its use.

The NBA or CNBA parameter specifies how many buffers are reserved for DEDB processing. This normal allotment is obtained at region startup and must be available from the total number specified for the control region. Although you can specify a number from 1 to 999, you coordinate the value across all BMPs and CCTLs that are to be concurrently active, so that the total can be specified by the DBBF parameter for the control region. If not enough buffers are available from the DBBF total, the BMP or CCTL cannot connect to the DBCTL environment.

After a CCTL is connected, each CCTL thread request for Fast Path PSBs receives an allotment of buffers.

The OBA or FPOB parameter specifies how many buffers are requested from the control region as overflow when the normal allotment (NBA, FPB) is depleted. The control region page fixes the largest number of overflow buffers specified for all

active BMPs and CCTL threads and makes these available to only one program at a time. Choose a value from 1 to 999 that is a suitable common value for all regions.

Both the NBA and OBA parameters are generated with a 0 (zero) value so that you must re-specify the individual values.

Related Reading: For more information on DEDB buffer considerations in a DBCTL environment, see *IMS/ESA Administration Guide: Database Manager*.

Online DEDB Utility Region Parameters in DCCTL or DB/DC

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

The procedure FPUTIL uses the first few positional parameters defined for the IMSFP procedure, but interprets several of the parameters as control for the online DEDB utilities. Table 14 shows these parameters.

Table 14. Generated Values for FPUTIL Procedure Parameters

FPUTIL Parameter	Generated Value	Purpose
IFP	IFP	Specifies Fast Path region
DBD	—	Specifies the DEDB name
—	DBF#FPU0	Utility program name
REST	00	Restart indicator
—	00	No overflow buffers
—	Null	Default startup, ask operator
—	1	Allows one abend
DIRCA	02	2 KB block for PCB
PRLD	Null	No preload
—	0	No time limit

The function performed by the utility depends on the input control statements. A series of DEDB areas can be scanned, have dependent segments deleted, or have the units of work reorganized. If the utility is to be restarted, the REST parameter is coded nonzero.

Online DEDB Utility Region Parameters in DBCTL

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

The procedure FPUTIL uses the first few positional parameters defined for the IMSFP procedure, but interprets several of the parameters as control for the online DEDB utilities. Table 15 on page 111 contains the defined parameters.

Table 15. Generated Values for FPUTIL Procedure Parameters

FPUTIL Parameter	Generated Value	Purpose
IFP	IFP	Specifies Fast Path region
DBD	-	Specifies the DEDB name
-	DBF#FPU0	Utility program name
REST	00	Restart indicator
-	00	No overflow buffers
-	null	Default startup, ask operator
-	1	Allow one abend
DIRCA	02	2K block for PCB
PRLD	null	No preload
-	0	No time limit

The function performed by the utility depends on the input control statements. A series of DEDB areas can be scanned, have dependent segments deleted, or have the units of work reorganized. If the utility is to be restarted, the REST parameter is coded nonzero.

Satisfying System Requirements for Data Propagation

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	
Note: The Data Capture exit routine is not available to CICS. DBCTL can use the exit routine, but only for BMPs.			

If your installation contains both IMS DL/I and DB2 relational databases, you can use the Data Capture exit routine to duplicate data between the two types of databases.

You can also use the DataPropagator Nonrelational (DPropNR) IBM licensed program to propagate data. If you use DPropNR, refer to the product documentation for details.

To use the Data Capture exit routine, your system must include the following minimum release levels:

- MVS/ESA Version 3 Release 1
- IMS TM Version 3 Release 1
- DB2 Version 2 Release 1

The Data Capture exit routine supports most IMS DB database structures, including:

- HDAM
- HIDAM
- HISAM
- SHISAM
- DEDB

Use of the Data Capture exit routine with these databases places some restrictions on the delete rules for logical-child segments.

Related Reading: For more information on delete rules, see *IMS/ESA Administration Guide: Database Manager*.

Defining the Data Capture Exit Routine

The Data Capture exit routine is defined in the DBD for the application program that requires data propagation. You must specify the exit routine name, exit routine data options, and the DBD source statements used to build the DBD execution-time blocks with the DBD Generation utility (DBDGEN).

Related Reading: For more information on DBDGEN, see *IMS/ESA Utilities Reference: System*.

You must also specify DB2 on the SSM= execution parameter for batch and online regions that contain application programs that make use of data propagation. You can do this at system definition or with JCL statements for batch application programs.

Running the Data Capture Exit Routine

The Data Capture exit routine is called when data segments are updated by an application program. Because the exit routine is defined for the DBD, it is called whenever a segment associated with the DBD is updated, regardless of the application program that updates the segment.

Your exit routine can be used to perform other database management functions, but if you update segments from within the exit routine, the changes are not propagated to DB2. The exit routine cannot call itself.

When the exit routine propagates data to DB2, the segments must be available in both the IMS and DB2 databases for the updates to be completed successfully. If any database segment is unavailable, a status code is returned to the I/O area of the DL/I call that attempted the update. You can use the inquiry (INQY) call to determine the system status while the application program is running.

Related Reading: For more information on the INQY call, see *IMS/ESA Application Programming: Database Manager*.

Storage Requirements for Data Capture

As your application program issues a DL/I call to update the database, the updates are stored as required for use by the Data Capture exit routine or the Asynchronous Data Capture. Because the amount of storage required can be significant for update functions like a cascade delete, a data space is acquired for each dependent region that uses the exit routine. The attributes of the data space vary for online and batch dependent regions, as illustrated in Table 16.

Table 16. Data Space Characteristics (Data Capture Exit Routine and Asynchronous Data Capture)

Attribute	Online Dependent Region	Batch Dependent Region
Number of data spaces	1 per dependent region	1
Data space name	SYSDFS01	@SYSDFS1
Storage key	Key 7, not fetch protected to allow access from dependent region in key 8	Key 8

Table 16. Data Space Characteristics (Data Capture Exit Routine and Asynchronous Data Capture) (continued)

Attribute	Online Dependent Region	Batch Dependent Region
Storage size	By region controller	By region controller. Default size used if space requested violates total size of key 8 data spaces.
Storage obtained	During region initialization	During region initialization if exit routines are defined.
Storage owned	By region controller TCB	By batch TCB
Added to access list	Dependent region address space, for access by program controller TCB in message regions. Control regions SAS address space for access by DL/I in an IMS DB/DC system when data capture is required. DEDB capture runs under program controller TCB.	Batch TCB
Deleted from access list	Dependent region always accessed. Deleted from control region SAS access list during thread termination if added to access list by data capture.	Not deleted.
Data space cleared	During normal thread termination for message regions if data space storage was referenced.	Not cleared.
Data space deleted	At region termination.	At MVS job termination.

You can control the use of data spaces with the IEFUSI SMF exit routine for key 8 batch regions. This exit routine determines the number and size of the data space available for key 8. If you have batch application programs that call the Data Capture exit routine, the data space specified for key 8 must be large enough to accommodate the data space requirements of data capture.

Related Reading: For more information on the IEFUSI SMF exit routine, see *MVS/ESA System Programming Library: System Modifications*.

Storage Failure

The two types of storage failure for data capture are:

- Data space not obtained. This type of error occurs in batch regions when a data space is not specified for each region. Online dependent regions can always obtain data space.
- Insufficient storage in the data space. In online dependent regions, storage space is specified by the region controller. Some database functions, such as cascade delete, require more than the space allocated for successful completion. Batch dependent regions can be limited in data space size. You must specify a data space large enough for data capture to complete successfully.

Either type of storage failure terminates the region with a U814 abend.

Related Reading: For more information on storage failure, see *IMS/ESA Failure Analysis Structure Tables (FAST) for Dump Analysis*.

Chapter 4. Establishing IMS Security

This chapter provides information to help you establish resource security for the IMS online system. It identifies resources that can be protected and the facilities available to protect them. It also gives design considerations and the steps you take to activate security.

In this Chapter:

Security Overview

When you initiate security safeguards, you need to balance requirements between those responsible for the security of resources and those users who legitimately need access to those resources. Because an individual assigned to resource security is held responsible for resources that might be compromised, that person should not allow easy access to dominate protection measures. On the other hand, users performing their assigned tasks need convenient access to the resources. The users and the security specialist should work out a balanced approach between the ease of resource access and the complexity of protecting that resource.

In an IMS system, two types of security exist. You can address one or both types:

- Securing the kind of resource to which a user has access. For example, a user might be allowed access to the Part database but not to the Customer Order database.
- Securing what the user can do to the resource after that user has access to it. For example, a user might be allowed to read a file but not to update it.

Resources That Can Be Protected

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Before you decide what security facilities to use in designing a secure IMS system, you should know which resources within the system need protection. In other words, you should decide what to protect before you decide how to protect it.

The following is a list of resources that can be protected:

IMS online system
The IMS system control program that enables online application programs to process the database through terminals.

PTERMs
Hardware devices attached to the computer and supported as a terminal by the IMS Transaction Manager. A physical terminal usually has one or more logical terminals associated with it.

LTERMs
A logical input or destination point in the system. Each logical terminal is associated with a static terminal or an ETO user.

Transaction
The process of executing a specific job as a result of input data or a message destined for an application program.

Command
A request from a terminal to perform a specific IMS service, such as altering a system resource status or displaying specific system information.

PSB Program specification block. The control block that describes a group of hierarchic databases and logical message destinations used by an online application program. (A PSB is composed of one or more PCBs.)

Online application program
A program in an IMS online system that performs work for a user. Message processing programs are activated by transactions from a terminal or by another application program. Batch message programs are activated by the dependent region controller after the region is started by JCL.

Database
A collection of data that is fundamental to the user's activity. Using a Program Communications Block (PCB), a program has a logical view of the database, as described by the IMS physical database design.

Dependent region
An area of storage in the IMS online system in which batch or online application programs are executed. The dependent region can be an MPP, IFP, or BMP region.

System data set
A collection of data that is fundamental to the operation of the IMS online system. An example is the IMS.MATRIX data set, which contains initialized security tables.

Security Choices Made during System Definition

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

You can make IMS security choices in any of three system definition macros: SECURITY, COMM, and IMSGEN.[5]The SECURITY macro lets you choose the type of security to be active in online execution. You can name the resources using the Security Maintenance utility alone, or you can also use RACF.

All security specifications are consolidated in the SECURITY macro. However, the COMM or IMSGEN keywords related to security specifications can also be used. The security-related keywords from these three macros are accepted hierarchically in the order SECURITY, COMM, and IMSGEN. If the SECURITY macro is used, the

5. The Security Maintenance utility and the Resource Access Control Facility (RACF) can also be used to implement security decisions. See *IMS/ESA Utilities Reference: System* for more information on these utilities.

specifications and defaults from that macro take precedence over any security specifications coded with COMM or IMSGEN.

Using the COMM or IMSGEN Macros for Security Options

The COMM macro and its keyword options control the same security functions as the IMSGEN macro.

The IMSGEN macro controls options of the password and terminal security functions of the Security Maintenance utility. The macro options permit the MTO to override the password and terminal security functions when restarting the IMS system using the /NRESTART command. The master terminal operator should be aware that any security override might compromise resource protection.

Default Terminal Command Security

If you do not specify any security type in any of the three macros, IMS system definition provides a basic level of resource security called default terminal security. *Default terminal security* permits a designated subset of commands to be entered from any non-master terminal. This basic security function is activated upon completion of stage 2 of IMS system definition. Default terminal security restrictions are removed by implementing LTERM security with the Security Maintenance utility (SMU) or RACF.

Default terminal security applies only to statically defined terminals. Terminals defined using ETO are automatically governed by an identical level of default security when the DFSCCMD0 exit routine is not used, or when the IMS DFSCCMD0 sample exit routine is used unmodified.

Deciding Which Security Facilities to Use

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

In choosing security facilities, you should consider the information presented here, as well as such things as your installation's security standards and operating procedures. Table 17 on page 118 summarizes the resources you can protect and the facilities you can use to protect them.

The following two examples show how you might decide which security facilities to use and how you could protect them.

- PTERM protection

  Assume your situation is as described in Table 17 on page 118, and that you have decided to limit the use of terminals A1, B1, and C1 to a subset of users by using signon verification. The user IDs and passwords are authorized by the Security Maintenance utility and an exit routine.

  You use the following Security Maintenance utility control and data statements to define this set of terminals (nodes) requiring signon verification:

  ```
  )(SIGN
    STERM A1
    STERM B1
    STERM C1
  ```

  Unless specified by additional Security Maintenance control and data statements, no other terminals are protected.

After defining which PTERMs require access, code a table of user IDs and passwords. This table can be link-edited into the IMS nucleus.

The Signon Verification exit routine, written by your installation, addresses this table for a match between the table and the user requesting access. An unmatched condition rejects the user.

- Transaction and command protection

  Assume you know which transactions or commands must be enabled through a terminal control point. Using LTERM security, you code the following Security Maintenance utility statements to describe the necessary set of transactions and commands:

```
)(TERMINAL     LTERM1
  TRANSACTION  ACCT
  TRANSACTION  DEBIT
  COMMAND      BROADCAST
  COMMAND      DISPLAY

)(TERMINAL     LTERM2
  COMMAND      DISPLAY
```

  The LTERM1 and LTERM2 sets authorize those specified transactions and commands to be entered through these terminals.

Table 17. Resources and the Facilities to Protect Them

Resources	Security Options/Type of Protection	Facilities
IMS online system	Extended resource protection (using APPL resource class)	RACF
System data set	OS password protection	MVS
	Data set protection (VSAM) (using PERMIT, RDEFINE classes)	RACF
Database	Segment sensitivity	PSBGEN RACF
	Field sensitivity	PSBGEN RACF
	Password security (for /LOCK, /UNLOCK commands)	Security Maintenance
PTERM[1]	Signon verification security	Security Maintenance and Exit Routine
	Signon verification security	RACF and Exit Routine
	Terminal-user security	RACF
	Password security (for /IAM, /LOCK, /UNLOCK commands)	Security Maintenance
LTERM[1]	Password security (for /IAM, /LOCK, /UNLOCK commands)	Security Maintenance
	Resource access security	Security Maintenance and Exit Routine
	Resource access security	Security Maintenance and RACF
Terminals defined with ETO	Signon verification security	RACF and Exit Routine
	Resource access security	RACF and Exit Routine
LU 6.2 inbound and IMS-managed outbound conversations	Allocate verification security	RACF and Exit Routine
	Resource access security	RACF and Exit Routine

Table 17. Resources and the Facilities to Protect Them (continued)

Resources	Security Options/Type of Protection	Facilities
PSB	Resource access security	Security Maintenance and Exit Routine
	Resource access security	Security Maintenance and RACF
	Resource access security	RACF and SAF[2]
Transaction	LTERM security[1]	Security Maintenance and RACF
	Resource access security	Security Maintenance and Exit Routine
	Resource access security	Security Maintenance and RACF
	Extended resource protection (using TIMS and GIMS classes)	RACF
	Password security[1]	
Command	Default terminal security	System Definition
	LTERM security[1]	Security Maintenance
	Password security[1]	Security Maintenance
	Transaction command security	Security Maintenance
	Extended resource protection (using CIMS and DIMS classes)	
Online application program	Password security (for `/IAM`, `/LOCK`, `/UNLOCK` commands)	Security Maintenance
	Extended resource protection (using APPL keyword)	RACF
Control region	Extended resource protection (using APPL resource class)	RACF
Dependent region	Resource access security	Security Maintenance and Exit Routine
	Resource access security	Security Maintenance and RACF
	Extended resource protection (using AIMS resource class)	

Notes:

1. Static terminals only. Not applicable to ETO-defined terminals.
2. Using the MVS System Authorization Facility (SAF) to secure PSBs applies to CPI-C driven applications only.

Design Considerations for IMS Security

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

This section explains how the various types of IMS security can be used. When you are deciding on each part of your security design, consider the physical actions that an end user must take to obtain access to the system. You will probably use more than one type of security checking. This section assumes:

- A user identification as a control point
- A logical terminal as a control point
- The master terminal as a control point

- The control of automated operator programs
- The use of RACF protection
- The use of a region as a control point

Limiting Access from a Terminal

Two approaches to access control for a terminal control point are:

- Have the end users identify themselves each time they use a terminal and authorize the set of transactions and commands each user needs.
- Associate with each LTERM in the network a list of transactions or commands eligible for entry.

Controlling Access by Signon Verification

Using signon verification, all terminals, or a subset of the terminals, require entry of a user identification as a parameter on a `/SIGN ON` command. That identification can also be associated with transactions and commands authorized to be issued from that terminal. To check signon verification as well as transaction and command verification, you can use the RACF licensed program, an exit routine, or both.

At system definition you specify, in the SECURITY macro, no signon verification, signon verification alone, or signon verification with transaction and command authorization. You can override these specifications with IMS procedure parameters, the `/NRESTART` command, or the `/ERESTART COLDSYS` command. You identify physical terminals that require signon verification through the Security Maintenance utility. Users who are defined through ETO are required to use signon verification. Because ETO is not supported by the Security Maintenance utility, signon verification can be performed through RACF or an equivalent security product, or through exit routines.

Note: Default terminal security with SMU does not apply to ETO. An equivalent level of security is provided for ETO terminals if DFSCCMD0 is not used for security or if the DFSCCMD0 sample exit routine is used unmodified.

Signon verification checks user IDs and passwords through the `/SIGN` command. It associates the user ID to the physical terminal from the time the `/SIGN ON` command is entered until the `/SIGN OFF` command is entered. Also, the user ID is included on all database update records on the system log caused by the processing invoked from the user's terminal.

Verification is done by RACF, by an exit routine, or by both. If you are using RACF, it checks for user ID, password, and protection group. If you are using an exit routine to implement signon verification, it can check for user ID and password.

You might decide to use only signon verification security, without RACF, to provide the basis for user accountability of database changes. In this case, the signon requirement should be defined for all terminals. In addition, you must provide a signon exit routine that has access to a list of all current user IDs.

The signon verification exit routine (DFSCSGNO) is described in "/SIGN ON/OFF Security Exit Routine" on page 140.

Authorizing Transactions and Commands

This section describes transactions and commands, and explains how to authorize them.

Transactions: Transaction authorization determines if a user ID is permitted to use a certain transaction. You can use an exit routine, RACF, or both to perform the transaction authorization. To gradually phase in RACF as your installation's transaction authorization method, use the Transaction Authorization exit routine. This routine can reject the transaction if the transaction is entered by an ETO terminal but is protected by SMU LTERM and password security. If you use both the Transaction Authorization exit routine and RACF, the Transaction Authorization exit routine is effective only after RACF has authorized the transaction or when the transaction is not defined to RACF. The Transaction Authorization exit routine (DFSCTRN0) is described under "Transaction Authorization Exit Routine" on page 140.

If you specify the REVERIFY option to RACF, the user must reenter the signon password with each transaction code. However, if you use the REVERIFY option and IMS password security on the same transaction, the RACF user password and the IMS password must be identical. Remember that the signon password is controlled by the user, whereas the IMS password is controlled by a security administrator, through the Security Maintenance utility. REVERIFY is not supported when a takeover occurs in an XRF complex. You might need to sign on again after a takeover if you are not class 1.

With program-to-program switching through the DL/I change (CHNG) call or by changing the transaction code in the SPA, you can use RACF, an exit routine, or both, to check transaction authorization. The same applies when the transaction code status is changed by the `/SET`, `/LOCK`, and `/UNLOCK` commands. In addition, as the application program associated with the transaction produces database changes, the user ID is logged with the change records on the IMS system log to identify the changes performed by a specific user.

Commands: When the Command Authorization exit routine (DFSCCMD0) is included in IMS.RESLIB, the exit routine provides a command authorization check. The Command Authorization exit routine can work in conjunction with RACF, or it can work independently, without using RACF. The Command Authorization exit routine is called for each IMS/ESA command. Default terminal security, RACF, or SMU password and terminal security is called first to perform the authorization. The return code is passed to the Command Authorization exit routine. DFSCCMD0 performs a final verification and determines the success or failure of the command authorization. DFSCCMD0 can also be used alone to perform the verification. If you want to establish a command authorization level more discrete than that provided by SMU or RACF, you can examine DFSCCMD0's input command buffer. DFSCCMD0's input command buffer contains the complete command stream.

If DFSCCMD0 exists in IMS.RESLIB, DFSCCMD0 is called for all device types including static, ETO, and LU 6.2.

Using LTERM Security

Another method of controlling the work performed through a terminal control point is to use the logical terminal name, which allows transactions or commands to be associated with an LTERM. As a result, those transactions or commands can be processed only from that LTERM. However, the LTERM is not otherwise restricted for data entry.

The Security Maintenance utility declarations support LTERM security with a transaction or command profile, or both, for each local or remote logical terminal. A profile is a list of transactions or commands that can pass through an LTERM. You

must explicitly declare commands or transactions assigned to an LTERM. An LTERM without a profile can be used to enter any unrestricted transaction or command.

LTERM security does not apply to ETO-defined terminals.

Designing LTERM Profiles

Table 18 suggests a possible approach to designing an LTERM profile. The first column shows the set of commands and transactions to be authorized for a specific LTERM. Password assignment to commands or transactions provides an additional level of security for LTERMs. Some examples of passwords are shown in the second column of the figure.

Use the following guidelines for passwords:

- Passwords should be six or more alphanumeric characters.
- Passwords should be nontrivial names not easily associated with a user.
- Passwords should be updated at intervals of 30 to 60 days.

Table 18. Designing an LTERM Profile

Command or Transaction	Optional Password
/CHANGE	C13579CC
/DELETE	D97531DD
/SET	S02468SS
ACCOUNT	(none)
PAY	XXOOXXOO
DEBIT	OOXXOOXX
OUTGO	(none)

A profile can be defined to include several users or to apply to several LTERMs, but, when assigned to an LTERM, security checking prevents any secured command or transaction not in the profile from passing through that control point. You can consider the profile as a way of limiting transactions or commands to a terminal, or as a way of making a terminal eligible for entry of a particular transaction or command. After the LTERM command-transaction profiles are completed, you can match them to the LTERM defined during IMS system definition. For an illustration of how to match LTERM names to profiles, see Table 19.

Table 19. Matching LTERM to Profiles

	System Defined Terminals				
Profiles	**LTERM 1**	**LTERM 2**	**LTERM 3**	**LTERM 4**	**LTERM 5**
PROFILE A				X	
PROFILE B					X
PROFILE C		X			
PROFILE D				X	X
PROFILE E			X		
PROFILE F	X				X

For example, an LTERM profile can be coded:

```
)( TERMINAL     LT36SST
   COMMAND      CHANGE
   TRANSACT     TACT234
   TRANSACT     TACT236
```

Or, with passwords:

```
)( TRANSACT     TACT234
   PASSWORD     S1332SST
   TERMINAL     LT36SST
)( COMMAND      CHANGE
   PASSWORD     S1332SST
   TERMINAL     LT36SST
```

This code would make only terminal LT36SST eligible for entry of TACT234, TACT236, and the /CHANGE command. The terminal LT36SST is not otherwise restricted.

Permitting Use of Commands by a Terminal User

If you do not specify other security facilities at system definition, default terminal security provides a basic protection for statically defined terminals by limiting the commands that can be entered through remote terminals.

Related Reading: For the commands that can be executed from the master terminal and from remote terminals if default terminal security is used, see *IMS/ESA Operator's Reference*.

Default command security allows certain IMS commands, called default-secured commands, to be issued only through the MTO and the system console. If a default-secured command is to be issued from another LTERM, it must be listed in the LTERM's command profile in the SMU input. If your system is to use them, you must define the command for RACF (or equivalent security product), the Command Authorization exit routine, or both.

IMS commands that can be entered by default from any terminal are called non-default-secured commands. If a non-default-secured command is listed in the SMU input, it becomes a secured command. No other LTERM (including the MTO and system console) can issue a secured command unless it is listed in the SMU input, with the command in the LTERM's command profile.

A further consideration for LTERM command profiles concerns their assignment to a PTERM. If a chain of LTERMs is associated with a terminal from which commands are entered, the first-in-chain needs a command profile. If that first LTERM becomes unavailable, others in the chain are not eligible to enter commands unless they, too, have command profiles defined in the Security Maintenance utility.

Using Password Protection with Command Keywords

To provide further verification for statically defined terminals before a command is accepted, you can require an accompanying password. The password is entered within parentheses immediately following the command verb. The password protection provided by the Security Maintenance utility gives you two capabilities for the use of command keywords:

- It prevents a change of status of the following resources by the /LOCK or /UNLOCK command unless a correct password is supplied:
  - Database
  - Program
  - LTERM or PTERM

- It prevents access from a remote, non-VTAM-switched terminal unless a correct password is supplied with the /IAM command.

Using the /DELETE command, the MTO can temporarily remove the password requirement for the remote PTERM. Signon verification protection for the PTERMs can be removed collectively by the MTO on restart. Using the /CHANGE command, the MTO can replace a current password with a new one. The actions of these commands and the security tables in effect at system startup remain in force until the next cold-start. Access through a PTERM control point can also be temporarily stopped by the MTO when disconnecting all LTERMs from a PTERM.

Security Considerations for the Master Terminal

The security of access from the master terminal is critical. Because the MTO can modify all security profiles during normal operations, you should consider protecting the terminal with a second level of control. Signon verification security provides this capability. The primary question is how much capability to modify security should be given to this second level of control.

Default terminal security does not and cannot prevent modifying the system's security profiles via the master terminal. LTERM security lets you specify whatever commands you want the MTO to be able to enter. The /ASSIGN, /CHANGE, and /DELETE commands are prime candidates to protect. At cold-start or warm-start time, the MTO can control the following security:

- Signon verification security
- Transaction authorization
- Terminal security
- Transaction command security
- Password security
- Command authorization

You can control these authorizations by coding the keywords shown in Table 20.

Table 20. SECURITY Macro Keywords

Enforced Security Option	Security Keyword	Parameter Value
Signon verification and transaction authorization	SECLVL	FORCSIGN FORCTRAN
Terminal security	TERMNL	FORCE
Transaction command security	TRANCMD	FORCE
Password security	PASSWD	FORCE

Security Considerations for AO Application Programs

Related Reading: For introductory information on automated operator (AO) applications, see *IMS/ESA Operations Guide*.

AO application programs can issue a subset of IMS operator commands. Because an operator command can compromise a security profile, you can prevent an AO application program from issuing sensitive commands that might modify IMS resources. The way in which you secure commands depends on whether the AO application program is using the DL/I CMD or ICMD calls to issue commands.

CMD Call

When AO application programs are using the CMD call to issue commands, transaction command security is used to protect security profiles. Transaction command security keeps an AO application program from issuing a command unless the transaction that invoked the program is authorized with a transaction-command profile in the Security Maintenance utility. You can use LTERM security to specify that the master terminal LTERM is the only eligible entry point for the transaction that invokes the AO application program. By using signon verification security and transaction authorization, you can limit the issuing of transaction codes to invoke AO application programs to a small set of users.

AO application programs can be viewed as pseudo-terminals, capable of issuing operator commands. Terminal security cannot be used to build a command authorization profile for a pseudo-terminal. Therefore, the Security Maintenance utility enables you to relate commands to a transaction by using TCOMMAND data statements with a CTRANS statement.

The following Security Maintenance utility input statements relate transactions to commands:

```
)(CTRANS    transaction code
  TCOMMAND  command
```

or

```
)(CTRANS    transaction code
  TCOMMAND  *
```

or

```
)(TCOMMAND  command
  CTRANS    transaction code
```

When an asterisk (*) is used in the TCOMMAND data statement, all commands in the command table DFSSMUL0 are used. This table is assembled and link-edited into IMS.RESLIB during system definition, and contains all the commands that can be issued by an AO application program. The following example allows the transaction PERS to be routed to an AO application program that is permitted to issue the `STOP` command (both forms are shown).

```
)(CTRANS   PERS
  TCOMMAND STOP
```

or

```
)(TCOMMAND STOP
  CTRANS   PERS
```

Related Reading: For the commands that can be issued by authorized transactions, see *IMS/ESA Operations Guide*.

When AO application programs issue a command using the CMD call, passwords are not used.

ICMD Call

When AO application programs are using the ICMD call to issue commands, commands can be secured using RACF (or the equivalent) or the Command Authorization exit routine (DFSCCMD0).

AOI security options are specified on the AOIS execution parameter in the DBC, DCC, and IMS procedures. AOIS lets you specify the following values:

A RACF (or equivalent) and DFSCCMD0 are to be called for ICMD command authorization. RACF is called first, then DFSCCMD0.

C DFSCCMD0 is to be called for ICMD command authorization.

N ICMD is not to be issued by any application program. N is the default.

R RACF (or equivalent) is to be called for ICMD command authorization.

S Skip command authorization; all application programs can issue ICMD calls.

Because the AOIS specification is not included in a checkpoint record, the specification can be changed each time IMS is initialized.

Related Reading: For more information on the AOIS parameter, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DFSCCMD0 is called during ICMD processing to perform command authorization checking (if AOIS=A or C is specified). DFSCCMD0 lets you secure commands issued in the ICMD call at the command verb, keyword, or resource name level. DFSCCMD0 must be linked into the IMS.RESLIB concatenation. The parameter list for DFSCCMD0 identifies who issued the command, whether RACF was called, and what the security code was.

Who Issued the Command:

- Terminal
- LU 6.2 application
- ICMD call, where user ID is used for command authorization
- ICMD call, where PSB name is used for command authorization

If RACF (or Equivalent) Was Called:

- SAF (System Authorization Facility) return code
- RACF return code
- RACF reason code

Security Code:

- X'80000000' RACF was not called (AOIS=C).
- X'00000000' User is authorized to RACF to issue command.
- X'00000004' RACF is not available.
- X'00000008' User is not defined to RACF.
- X'0000000C' Command is not protected by RACF.
- X'00000010' User is not authorized to issue command.

Related Reading: For more information on DFSCCMD0, see *IMS/ESA Customization Guide* .

With RACF, you can specify access authority under which system resources are made available to users of the system. The resources defined to RACF for AOI are the commands that can be issued in the ICMD call. Users who can issue a command are defined to RACF based on the execution mode of the AO application program issuing ICMD. The modes of execution and the associated user, defined to RACF, follow. If RACF security is used, only those AO application programs executing with a user ID defined to RACF are allowed to enter commands.

MPP or IFP If a message GU call has completed, user ID is used to determine whether the user can issue

commands using ICMD. User ID is of a signed-on terminal or the LTERM name of the signed-off terminal where the transaction is issued. If GU is not issued, PSB name is used.

BMP

If a message GU call has completed, user ID is used to determine whether the user can issue commands using ICMD. User ID is of a signed-on terminal or the LTERM name of the signed-off terminal where the transaction is issued. If GU is not issued or if the BMP is non-message driven, the value of the USER parameter specified on the JCL JOB statement is used. If the USER parameter is not specified, a user ID of 0000000 is used.

DRA THREAD

The security token passed in the PAPL for a schedule request is used to determine whether the user can issue commands using ICMD.

BMP regions are initiated by a /START command or by submitting JCL. If you use a /START command, you should use RACF to protect the library containing the cataloged procedures. If you use JCL, BMP startup security is provided by SMU and RACF, or by SMU and an exit routine. You control access to the BMP region by defining a unique user ID to RACF on the JOB card and specifying the AGN (application group name) parameter on the EXEC statement.

When AO application programs issue a command using ICMD, passwords are not used.

Related Reading: For the commands that can be issued by AO application programs, see *IMS/ESA Operator's Reference*.

Security Considerations for Fast Path Application Programs

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X
Note: The DCCTL environment does not support Fast Path databases. It does support Fast Path processing and transactions.			

When you design security protection for Fast Path application programs, or for DL/I programs that access Fast Path databases, you should consider:

- The use of LTERM security for all terminals taking part in message-driven application programs.
- The protection of the network of Fast Path terminals by signon verification and transaction authorization.
- The protection of the processing in a dependent region by resource access security: assign an AGN to the region and authorize a PSB.

Security Considerations for CPI-C Driven Application Programs

You can secure any PSB specified on an APSB call from a CPI-C driven application program using the MVS System Authorization Facility (SAF). APSB SAF security overrides APSB AGN Table security. If you use RACF, you must use RACF 1.9.2 or later on your MVS system to gain the benefits of APSB SAF security.

Security Design Considerations

Once APSB SAF is security-enabled, IMS calls SAF to secure the PSB specified on an APSB call against the AIMS or Axxxxxxx general resource class (where xxxxxxx is the value specified on the RCLASS= parameter of the IMS SECURITY macro) based on the USERID of the user associated with the CPI-C application. Therefore, you must define the PSBs that you want protected by RACF (or your installation exit) to the AIMS or Axxxxxxx resource class. Since the AIMS resource class can contain both PSBs and AGNs, all PSB names and AGNs specified in the AIMS resource class should be unique. In addition, you must specify `RCLASS=IMS|xxxxxxx` **and** `TYPE=RACFAGN|RACFTERM` on the IMS SECURITY macro at IMS system generation time.

Also, when APSB SAF Security is enabled, if the PSB is not defined to the AIMS resource class or the AIMS resource class is not active, the PSB is secured using AGN Table security if Resource Access Security (RAS) is active.

Related Reading: For more information on RACF and IMS, see the "RACF and IMS" chapter in *RACF Security Administrator's Guide.*

Use of the RACF Data Space

When the RACF (or an equivalent product) data space is supported (RACF 2.1 or later), IMS loads the RACF profiles for the IMS commands and transactions into that data space instead of the IMS control region.

Use of the RACF data space invalidates the IMS online change support for RACF with the `/MODIFY` command. The IMS online change support is still valid, though, when the RACF data space is not being used.

The message `DFS3432 RACF PARAMETER INVALID IF RACF DATA SPACE USED` is issued if the RACF parameter is used on the `/MODIFY PREPARE` command when the RACF data space is being used. You can use the RACF command `SETROPTS RACLIST (classname) REFRESH` to refresh the RACF resource profiles in the RACF data space without requiring the IMS applications to suspend work.

Planning for Security in a Shared-Queues Environment

Because IMS can act as either a front-end subsystem, a back-end subsystem, or both, the system programmer must consider security based on the role that the IMS subsystem is playing. The two environments in which front-end processing can be split from back-end processing are multiple systems coupling (MSC) and shared queues.

Front-End Security

Using SMU to secure commands for an LTERM provides security only for the local IMS front end. Using SMU to secure transactions for an LTERM provides security only for the IMS on which the security check is being made, and only if the resources are defined to that IMS subsystem. Using SMU for password security for an IMS resource provides security only for the IMS subsystem on which the security check is being made, and only if the resources are defined to that IMS subsystem.

Back-End Security

For the IMS back end, you can use RACF (or an equivalent product) or the SMU for security. `CHNG` and `AUTH` calls and IMS conversational deferred program switches (that occur in the same IMS as the inputting terminal) perform an authorization check to determine whether the user who entered the transaction is authorized to use the IMS resource.

To perform a RACF authorization from the dependent region, a security environment first must be established. If the dependent region is part of the same IMS as the inputting terminal, and the user who entered the transaction is still signed on, then the security environment that is created in the IMS control region at signon is used for the authorization call. The security environment that is created at signon is not available under the following conditions:

- The dependent region is part of the same IMS as the inputting terminal, but the user has signed off.
- The dependent region is part of another IMS, connected to the IMS with the inputting terminal by an MSC link.
- The dependent region is part of another IMS in a Sysplex with IMS shared message queues support.

In these conditions, the security environment must be dynamically created in order to perform the RACF authorization check. Dynamic creation of this security environment increases the time required to process a `CHNG` or `AUTH` call. The dynamically created security environment is kept until IMS is done with the message (until the next syncpoint or `GET UNIQUE`). Use the user exit DFSBSEX0 to control when IMS performs the dynamic creation of the security environment.

Related Reading: For an explanation of DFSBSEX0 see *IMS/ESA Customization Guide*.

For input from APPC or OTMA, if security is defined as FULL, the security environment has already been created before any `CHNG` or `AUTH` call. If APPC/OTMA security is defined as NONE, RACF is not called. If APPC/OTMA security is defined as CHECK, and a `CHNG` or `AUTH` call is made, a dynamic security environment has to be created.

When the following IMS exit routines are called because of an application program `CHNG` or `AUTH` call, the address of the CTB is zero when the call is made from an IMS dependent region that is not part of the same IMS as the inputting terminal:

- Command Authorization Exit Routine (DFSCCMD0)
- Transaction Authorization Exit Routine (DFSCTRN0)
- Security Reverification Exit Routine (DFSCTSE0)

AGN security for LTERMs in a shared-queues environment is supported only for LTERMs that are statically defined to that IMS back end.

Using SMU to secure transactions for an LTERM provides security only for the IMS on which the security check is being made and only if the resources are defined to that IMS. IMS back ends can use SMU LTERM security for the `CHNG` call and for deferred conversational program switches.

In general, SMU can be used for security only if the control blocks that are required for the security check are local to the IMS making the check.

Reverifying the User on Entry of a Command or Transaction

IMS provides the RVFY= parameter in the IMS procedure for customers who want to force reverification that the operator who signed on to a terminal is the same operator who is now entering a command or transaction. This reverification is done with RACF by including the word `'REVERIFY'` in the APPLDATA field of the command or transaction profile. For example:

```
RDEFINE Txxx tran-name UACC(NONE) APPLDATA('REVERIFY')
```

Each time the user enters this transaction code, the RACF password must be entered where an IMS password would be entered if the transaction were password protected.

Using RACF to Protect Physical Terminals

RACF offers a terminal-user security function that ranges from no security for a particular terminal to permitting a certain predefined list of users access through a physical control point. A terminal-user profile can be created for every PTERM in the IMS system. See Table 21 for an example of a terminal-user profile.

Table 21. Terminal-User Profile

User	Physical Terminal				
	PTERMA	PTERMB	PTERMC	LTERMD	LTERME
USER #1	X				X
USER #2	X	X			
USER #3				X	X
USER #4					X
USER #5		X			X

Users defined through ETO use signon verification to gain access to IMS transactions or commands. Because ETO does not use SMU, dynamic terminal security must be defined through a security product such as RACF or through exit routines.

Signon Password Reverifying with RACF

Using signon verification security, a check of the password is made with entry of the `/SIGN` command, by either the Signon Verification exit routine or by RACF. If RACF is used, the `/SIGN ON` command user ID must be accompanied by a user password and, optionally, by other signon data. If the RACF reverification option has been specified, the password is saved so it can be compared with the password reentered with the transaction code.

RACF Password Protection

Password protection as defined by RACF supersedes the capabilities defined by the Security Maintenance utility. RACF passwords are defined and maintained by the user, whereas IMS passwords are defined and maintained by the security administrator, who uses the Security Maintenance utility. After the RACF resource class is initialized with a password, the user can change its value. If signon verification is provided by an exit routine and not by RACF, the table of user IDs and passwords must be changed by another link-edit into the IMS nucleus. However, for ETO signon verification, the table of user IDs and passwords does not need to be changed by another link-edit into the IMS nucleus. If the table is loaded by exit routine DFSINTX0 or if it is part of exit routine DFSSGNX0, IMS does not need to be restarted in order to have the table refreshed.

Considerations for APPC/IMS Security

APPC/MVS does not verify user authority to access transaction codes or specific IMS systems. To provide complete security verifications of a user's authority to execute transactions with LU 6.2 devices, you must use RACF. APPC/MVS uses RACF resource class APPCTPX for security. This holds a profile for every IMS transaction defined to RACF for transaction authorization verification. Exit routine DFSCTRN0 is called for transactions, and DFSCCMD0 is called for commands.

Security Considerations for ETO

ETO provides dynamic LTERM support. You can dynamically create and allocate local LTERMs to a terminal session based on user signon or the application ISRT process. An LTERM can be associated with a specific user ID instead of a physical terminal. By associating an LTERM with a user ID, you prevent the wrong user from receiving messages at a physical terminal.

Related Reading: For more information on ETO security, see *IMS/ESA Administration Guide: Transaction Manager*.

The initialization exit routine (DFSINTX0) can cause user-defined security information to be loaded and made available to the following security exit routines. IMS searches for and uses these security facilities in the following order:

Exit Routine	Description
1. DFSLGNX0	You can include this Installation Logon exit routine in IMS.RESLIB when ETO is implemented. Available functions include accepting or rejecting a session, locating logon user data, extracting associated printer LU names, creating a parameter list containing printer LU names, and passing the address of the parameter list to IMS.
2. DFSSGNX0 [6]	You can include this installation exit routine in IMS.RESLIB when ETO is implemented. Available functions include ACCEPT/REJECT SIGNON[7], multiple sessions, LTERM additions, removing extraneous keywords, telling IMS to override the TERMINAL macro's MSGDEL parameter and TRANRESP option for a specific user,[8]locating the parameter list created by DFSLGNX0, creating user names algorithmically from user IDs for each associated printer[9], and passing the printer parameters to IMS. You can also use the DFSSGNX0 exit routine to dynamically create node user descriptors and to create a user structure name that is completely different from both the user ID having a suffix and the node name.

Related Reading: For more information on using the DFSINTX0, DFSLGNX0, and DFSSGNX0 exit routines, see *IMS/ESA Customization Guide*.

Limiting Access from a Dependent Region

Another resource to protect is the dependent message region. You can protect this resource by preventing the start of an unauthorized dependent message region by the start-of-task JCL and by preventing the use of unauthorized resources in a dependent region. Resource access security provides this protection.

You can authorize the following resources to be used by dependent regions:

---

6. In addition, you can also include functions in DFSSGNX0 that are specific to your installation and apply to static terminals. See *IMS/ESA Customization Guide* for additional information.
7. This function can also be used for static terminals.
8. You can also use the Output User Creation exit routine (DFSINSX0) for this purpose.
9. See *IMS/ESA Administration Guide: Transaction Manager* for more information about associated printing.

- BMP regions
  - Transaction codes
  - PSBs
  - Logical terminals
- MPP regions
  - Transaction codes
  - Logical terminals
- IFP regions
  - PSBs
  - Logical terminals

In order to control the application programs that are authorized to use a dependent region, group them under an application group name (AGN). The JCL that starts up a dependent region stipulates which group of application programs are eligible to be scheduled, if the AGN is known to the control region. Further validation restricts that region's use to PSBs, transaction codes, or LTERM names associated with the AGN.

The use of the AGN ties the two-part protection together. The dependent region and resource authorization is implemented by the Security Maintenance utility and an exit routine, or by the Security Maintenance utility and RACF. You can prevent an unauthorized dependent region from starting by performing the following four steps:

1. Code the AGN parameter in the EXEC statement of the dependent region JCL. The AGN parameter name associated with the JCL is the same name associated with the profile of the limited resources permitted to use the dependent region.
2. Use the Security Maintenance utility to create an entry in the AGN table, with the same name used by the AGN parameter on the JCL. Then define the resources permitted to use the dependent region. This step alone provides the second part of resource access security.
3. Use the Security Maintenance utility and an exit routine to authorize the region. The exit routine must compare the AGN entries in the application group name table to the AGN name associated with the start-of-task JCL. A mismatch prevents starting the dependent region. For the Security Maintenance utility and RACF, a different user ID is assigned to the job cards of all dependent region JCL. These same user IDs are entered into the RACF system.
4. Create an entry in the class descriptor table (CDT) for the AGN resource for RACF. If you choose not to take the default description, the ICHERCDE macro creates the entry in the CDT. All AGN names assigned to dependent region JCL are logically related to the CDT resource entry by the RACF RDEFINE statement. RACF connects the user ID to its appropriate AGN source. RACF prevents starting a dependent region if the user ID of the start-of-task JCL is not permitted.

In CPI-Communications-driven application programs, resource access security can restrict the use of a PSB by using the AGN. If the PSB is an unauthorized resource, IMS rejects the DL/I allocate PSB (APSB) call but does not abend the program.

The ISIS parameter on the IMS control region EXEC statement controls the use of resource access security. If a value of 2 is specified, an exit routine checks the request to use the dependent region. If a value of 1 is specified, RACF is used instead of an exit routine.

For CHNG and AUTH calls and IMS conversational deferred program switches that occur in the same IMS as the inputting terminal, IMS performs an authorization check to determine whether the user who entered the transaction is authorized to use the IMS resource. You can have IMS use RACF (or an equivalent product) or the SMU for the authorization check.

Associating Resources with Application Group Names

To define the resources authorized for dependent regions, you must name the application groups and their authorized logical terminals, PSBs, and transaction codes. You can define up to 5000 LTERMs, AGNs, and TRANS by means of resource access statements as shown in “Examples of Security Maintenance Utility Input Statements” on page 136. There is no limit to the number of PSBs that can be defined per AGN.

If a transaction is selected to be scheduled in an MPP region, that transaction code must have been declared through the Security Maintenance utility as valid for the associated AGN. Otherwise, the scheduling of that PSB is rejected. For BMPs, the Security Maintenance utility declaration can specify one or more of PSB name, transaction code, or the output LTERM. If the declared entries do not match, the transaction abends. For a Fast Path region, the PSB name declared must match the AGN; otherwise, the transaction abends.

Designing a Resource-Access Exit Routine

The Resource-Access exit routine (DFSISIS0) authorizes starting a dependent region. A region starts when a match occurs between a name entry in the application group name table and the name assigned to the AGN keyword of the EXEC statement in member IMSBATCH or IMSMSG of IMS.PROCLIB. When the IMS control program calls this exit routine, two registers contain pointers to the sources that are to be compared by this exit routine. The DFSISIS0 routine supplied with the system must be replaced. (The routine supplied refuses authorization to all callers by notifying the IMS control program that an invalid dependent region has tried to start.) The use of the exit routine is specified by coding the AGNEXIT parameter for the TYPE keyword in the SECURITY macro.

Related Reading: For more information on register usage, see *IMS/ESA Customization Guide*.

Authorizing Resource Use in a Dependent Region

For the IMS dependent region, you can use RACF (or an equivalent product) or the SMU for security. CHNG and AUTH calls and IMS conversational deferred program switches that occur in the same IMS as the inputting terminal perform an authorization check to determine whether the user who entered the transaction is authorized to use the IMS resource.

To perform a RACF authorization from the dependent region, a security environment must be established. If the dependent region is part of the same IMS as the inputting terminal and the user who entered the transaction is still signed on, then the security environment that is created in the IMS control region at signon is used for the authorization call. The security environment that is created at signon is not available under the following conditions:

- The dependent region is part of the same IMS as the inputting terminal, but the user has signed off.
- The dependent region is part of another IMS, connected to the IMS with the inputting terminal by an MSC link.
- The dependent region is part of another IMS in a Sysplex with IMS shared message queues support.

In these conditions, the security environment must be dynamically created to perform the RACF authorization check. Dynamic creation of this security environment increases the time required to process a CHNG or AUTH call. The dynamically created security environment is kept until IMS is done with the message (that is, until the next syncpoint or GET UNIQUE).

For input from APPC or OTMA, if security is defined as FULL, then the security environment is already created before any CHNG or AUTH call. If APPC or OTMA security is defined as CHECK and a CHNG or AUTH call is made, then a dynamic security environment has to be created.

The user ID that is used for creating the security environment for making the authorization call is based on the following environments and criteria:

MPP or IFP
: If a message GU call has completed, then the security value from the input message is used to perform the authorization. The security value is the user ID of a signed-on terminal or the LTERM name of the signed-off terminal where the transaction is issued. If a GU has not been issued, then the PSB name is used.

BMP
: If a message GU call has completed, then the security value from the input message is used to perform the authorization. The security value is the user ID of a signed-on terminal or the LTERM name of the signed-off terminal where the transaction is issued. If a GU has not been issued or if the BMP is non-message driven, then the value of the USER= parameter that is specified on the JCL JOB statement is used. If the USER= parameter is not specified, then a user ID of 0000000 is used.

If no ACEE exists in the IMS control region and a dynamic security environment cannot be created dynamically, then a default security environment is used. If an IMS BMP has PARDLI=1 specified or an IMS system is specified with LSO=Y, then the default security environment is the environment of the IMS control region that is created with the user ID that is associated with the IMS control region. Otherwise, the default security environment is that of the IMS dependent region that is created with the user ID that is associated with the IMS dependent region.

When the following IMS exit routines are called because of an application program CHNG or AUTH call, the address of the CTB is zero if the call is made from an IMS dependent region that is not part of the same IMS as the inputting terminal:

- Command Authorization exit routine (DFSCCMD0)
- Transaction Authorization exit routine (DFSCTRN0)
- Security Reverification exit routine (DFSCTSE0)

Using SMU to secure transactions for an LTERM provides security only for the IMS on which the security check is being made and only if the resources are defined to that IMS. IMS back ends can use SMU LTERM security for the CHNG call and for deferred conversational program switches.

In general, SMU can be used for security only if the control blocks that are required for the security check are local to the IMS that is making the check.

Activating IMS Security

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

This section presents guidance on the steps you take to activate your IMS security design, using the Security Maintenance utility, RACF, and program exit routines. Depending upon the security facilities you choose to use, you need to perform one or more of the tasks described in the following sections:

- "Defining the SECURITY Macro"
- "Coding the TYPE Keyword"
- "Preparing to Use the IMS Security Maintenance Utility" on page 136
- "Preparing Exit Routines as Part of Authorization" on page 140
- "Preparing to Use RACF for Security" on page 140

Defining the SECURITY Macro

The purposes of the macro keywords are described below.

Related Reading: For guidance on how to specify the keywords, see the section on the SECURITY macro in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Coding the TYPE Keyword

Using this keyword, you specify which security facility, or combination of facilities, is to be included in your system. The parameter values to invoke security checking are:

SIGNEXIT
For signon verification security. All terminals, or a subset of the terminals, require entry of a user ID as a parameter on a `/SIGN ON` command. User IDs are to be checked by an installation-written exit routine.

TRANEXIT
For transaction authorization. An installation-written exit routine authorizes transactions. If you do not specify RACFTERM, the signon exit routine must be in place; that is, SIGNEXIT is assumed.

AGNEXIT
For resource access security. An installation-written exit routine checks which programs are authorized under the AGN to use the region. Further validation restricts that region's use to PSBs, transaction codes, or LTERM names associated with the AGN.

If you plan to use the RACF licensed program for your MVS system, you specify:

RACFAGN
For use of a RACF exit routine to verify AGN validity. The AGN names are defined as a protection group to RACF.

RACFTERM
For use of a RACF exit routine to verify signon data or transaction authorization. The user identification and password are defined to RACF. The identifications can be a protection group.

RACFCOM
This keyword determines whether RACF is used to verify command authorization for terminals. If RACFCOM is specified, RACF is called to validate command authorization.

Additional keywords control system security options or functions within RACF or SMU.

RCLASS This keyword identifies the IMS system as a resource class to RACF. You can then use RACF for signon verification, transaction authorization, or both. Resource class assignment is explained in Table 22 on page 142.

SECCNT This keyword controls the number of security violations allowedbefore the master terminal operator (MTO) is notified. You can specify that no notification be sent, or that notification be sent each time 1, 2, or 3 violations occur for a physical terminal or a transaction. For ETO, even when you specify that the MTO is to be notified each time 2 or 3 violations occur, the MTO is to be notified each time only 1 violation occurs for an ETO terminal.

SECLVL, TERMNL, TRANCMD, PASSWD
These keywords control the amount of security checking flexibility given to the MTO for the current online execution.

Preparing to Use the IMS Security Maintenance Utility

The first step toward activating the Security Maintenance utility is to provide input control and data statements. A control statement names the resource to be protected and the data statement names the security to be established for the named resource.

Related Reading: For the rules for coding the Security Maintenance utility, see *IMS/ESA Utilities Reference: System*.

This section contains:

Note: SMU is not available for ETO terminals.

Examples of Security Maintenance Utility Input Statements

The examples below illustrate various aspects of SMU input statements.

Example 1: The following are examples of passwords assigned to each program:

```
)( PROGRAM     ACCT
    PASSWORD   DOLLAR

)( PROGRAM     ENG560
    PASSWORD   PARTNO

)( PROGRAM     LOGREC
    PASSWORD   NONE

)( PROGRAM     AGC0568
    PASSWORD   MONEY
```

Example 2: The following are examples of passwords assigned to each database:

```
)( DATABASE    ACCTLOG
    PASSWORD   LOG

)( DATABASE    ACCTREC
    PASSWORD   REC

)( DATABASE    PARTSREC
    PASSWORD   PIERSQ

)( DATABASE    PARTSREC
    PASSWORD   ASSY
```

Example 3: The following are examples of passwords assigned to commands:

```
)( COMMAND     CHANGE
    PASSWORD   PSWD1

)( COMMAND     PURGE
    PASSWORD   PSWD2
```

Example 4: The following are examples of lists of terminals that can use each transaction code, with common passwords assigned to groups of transaction codes:

```
)( TRANSACT    ACCTCHG
    PASSWORD   CHARGE
    TERMINAL   A875111
    TERMINAL   C8751112
    TERMINAL   D8751113

)( TRANSACT    ACTY
    PASSWORD   GO
    TERMINAL   A8751111
    TERMINAL   A8751112

)( TRANSACT    TNL
    PASSWORD   QTY
    TERMINAL   DEPT650
    TERMINAL   DEPT610
    TERMINAL   DEPT620
    TERMINAL   DEPT631
```

Example 5: The following are examples of commands and transaction codes that can be entered from the master terminal:

```
)( TERMINAL    MASTER
    TRANSACT   ACCTCHG
    TRANSACT   ACTY
    TRANSACT   TNL
    TRANSACT   INQUIRY
    TRANSACT   INQ
    TRANSACT   ENG
    TRANSACT   ACCT
    COMMAND    BROADCAST
    COMMAND    START
    COMMAND    STOP
    COMMAND    PSTOP
    COMMAND    PURGE
    COMMAND    CHANGE
    COMMAND    DELETE
    COMMAND    ASSIGN
    COMMAND    CHECKPOINT
    COMMAND    DBDUMP
    COMMAND    NRESTART
    COMMAND    ERESTART
```

```
    COMMAND    DBRECOVERY
    COMMAND    IDLE
    COMMAND    RSTART
    COMMAND    DISPLAY
```

Example 6: The following is an example of signon verification for all terminals:

```
)(  SIGN
    STERM ALL
```

Example 7: The following is an example of transaction command security for certain commands:

```
)(  TCOMMAND   STOP
    CTRANS     ADDINV
    CTRANS     APOL11
    CTRANS     APOL12
)(  CTRANS APOL13
    TCOMMAND COMPT
```

Example 8: The following is an example of transaction command security specifying unlimited command use for an automated operator program:

```
)(  CTRANS     AUTOCTL
    TCOMMAND   *
```

Example 9: The following is an example of resource access security for several regions:

```
)(  AGN   TEST001
    AGPSB DDLTBP01
    AGTRAN TRAN13C0
    AGLTERM DD3270L4
)(  AGN   TEST002
    AGPSB ALL
    AGTRAN ALL
    AGLTERM ALL
)(  AGN   TEST003
    AGPSB APOL1
    AGPSB A3270
    AGPSB GISBMP09
    AGTRAN ADDINV
    AGTRAN APOL15
    AGLTERM TERM0001
    AGLTERM TERM0002
    AGLTERM TERM0003
    AGLTERM TERM0004
)(  AGN   TEST004
    AGPSB A3270
    AGTRAN ADDINV
)(  AGN   TEST005
    AGPSB INTCON
```

Executing the IMS Security Maintenance Utility

The Security Maintenance utility is run after completion of each system definition because the internal system description blocks, created by the system definition process, are used as input to the Security Maintenance utility. The utility is run for every modification of the existing resource access profiles and when a new version of the security tables in IMS.MATRIX is required. The SECURITY member in IMS.PROCLIB contains the required JCL to read the input Security Maintenance utility control and data statements and to create the security tables in IMS.MATRIX. You can modify the procedure JCL to match the input data device type. MVS password or RACF data set protection should be used to protect IMS.PROCLIB and the data set containing the Security Maintenance utility control and data statements.

The Security Maintenance utility runs as a three-step job. The first step accepts the input control and data statements and checks them against the IMS system being maintained to ensure correct format and validity. When no errors are detected in the first step, the second step, an operating system assembly, is performed. Step 3 is a link-edit that takes the assembly output from step 2 and creates the following:

- Signon table (DFSISSOx)
- Communication terminal matrix (DFSISTBx)
- Terminal offset list (DFSISTLx)
- Transaction offset list and table (DFSISTTx)
- Communication password table (DFSISPBx)
- Password offset list (DFSISPLx)
- Transaction command matrix (DFSISTCx)
- Application group name table (DFSAGT0x)

Depending on the input presented, a variable number of output load modules are created as members of the IMS.MATRIX data set. These members cannot be reprocessed using the linkage editor.

Allocating the IMS.MATRIX Data Set

The maximum size of any generated matrix in bytes (M) is:

```
M = (I X R)/8
```

- Where, for the terminal matrix:

  I The total number of LTERMs referenced in the Security Maintenance utility input. In order to produce a valid terminal matrix, the number of LTERMs specified at system generation cannot exceed 65535.

  R The total number of transactions + commands associated with the LTERMs.

- Where, for the password matrix:

  I The total number of unique passwords in the Security Maintenance utility input.

  R The total number of resources associated with a password.

- Where, for the transaction-command matrix:

  I The total number of transactions issuing commands.

  R The total number of command verbs associated with transactions.

Controlling Versions of the Security Matrix Tables

Additional versions of the Security Maintenance utility security tables are created with each run of the security procedure. Eight security table members in IMS.MATRIX carry a suffix version code.

The suffix version codes are controlled by the IMS symbolic parameter in the EXEC statement. This alphanumeric parameter must match the alphanumeric suffix of the IMS nucleus used in the next IMS restart. The OPTN symbolic parameter in the PARM keyword of the Security Maintenance utility EXEC statement controls the update of the current security tables.

If you want to verify the resource entries before updating the security tables, specify LIST as a parameter value to check validity and list the new security tables. The UPDATE parameter checks the validity of, lists, and updates the security tables in IMS.MATRIX. After the update occurs, this version is not used until the next IMS restart or until an online change activates it. The last version of security tables is

resident in main storage as a result of the previous system restart. On every Security Maintenance utility update run, the last security tables created in IMS.MATRIX are overwritten by the current version of the security tables. To save more than one version of the security tables, you must add JCL to the security procedure. Alternatively, when IMS.MATRIX is used as a staging library, you can copy it using the Online Change utility.

Preparing Exit Routines as Part of Authorization

Prepare the following exit routines as part of authorization:

- /SIGN ON/OFF security exit routine
- Transaction Authorization exit routine
- Command Authorization exit routine

/SIGN ON/OFF Security Exit Routine

The /SIGN ON/OFF Security exit routine must be coded by your installation as module DFSCSGN0. This exit routine should have access to a table of valid user IDs and their associated passwords. For addressability, the table should reside in module DFSCSGN0, the Transaction Authorization exit routine (DFSCTRN0), or in the IMS nucleus. The exit routine should note each successful signon. When the `/SIGN OFF` command is executed, the exit routine should mark that user ID available for /SIGN ON. The exit routine can place information in the data portion of the user verification string for logging. (An address in a register points to the user verification string.)

Related Reading: For more information on register usage and the /SIGN ON/OFF Security exit routine, see *IMS/ESA Customization Guide*.

Transaction Authorization Exit Routine

The Transaction Authorization exit routine must be coded by your installation as module DFSCTRN0. This exit routine should have access to a table of valid user IDs, passwords, and transactions associated with each valid user ID. For addressability, this table should reside in module DFSCTRN0, the /SIGN ON/OFF Security exit routine (DFSCSGN0), or in the IMS nucleus. If the table is in the nucleus, it can be shared by the Transaction Authorization exit routine and the Signon Verification exit routine.

If you use message edit routines, security is checked after the message is edited.

Related Reading: For more information on register usage and the Transaction Authorization exit routine, see *IMS/ESA Customization Guide*.

Command Authorization Exit Routine

The Command Authorization exit routine must be coded by your installation as module DFSCCMD0. This exit routine should have access to a table of valid user IDs, passwords, and commands associated with each valid user ID. For addressability, this table should reside in module DFSCCMD0, the /SIGN ON/OFF Security exit routine (DFSCSGN0), or in the IMS nucleus. If the table is in the nucleus, it can be shared by the Command Authorization exit routine, the Transaction Authorization exit routine, and the Signon Verification exit routine.

Related Reading: For more information on register usage and the Command Authorization exit routine, see *IMS/ESA Customization Guide*.

Preparing to Use RACF for Security

You can implement a security plan using RACF by performing these steps:

1. Prepare a list of all the IMS online resources to be protected, arranging them in groups to give an overview of the total resources covered.
2. Select the security facilities that protect the resource groups.
3. Design and list resource access profiles.
4. Design screen formats to include non-display fields for passwords in transactions and commands.
5. Code the SECURITY, COMM, and IMSGEN macros.
6. Code the resource access profiles, using the Security Maintenance utility.
7. Describe the system resource classes to RACF.
8. Add users, groups, and data sets to RACF.
9. Execute the Security Maintenance utility.
10. Define transactions and transaction groups to RACF.
11. Define databases, segments, fields, and other resources and resource groups to RACF.
12. Define commands and command groups to RACF.
13. Define extended resource protection sources (APPL).
14. Modify JCL procedures in IMS.PROCLIB.

A variety of RACF resource classes are used by the IMS security function. These classes define individual resources or groups of resources, and they are divided into eight categories:

Transaction
The transaction resource class holds a profile for every IMS transaction defined to RACF for transaction authorization checking. The transaction group resource class allows grouping of IMS transactions that have a common access authority profile.

Command
The command resource class contains a profile for every command defined to RACF for command authorization checking. The command group resource class allows grouping of IMS commands that have a common access authority profile. Commands are defined for authorized user IDs.

Application
The application group resource class holds a profile for every AGN and PSB defined to RACF. The application resource class holds a profile of every subsystem defined to RACF. The IMS system is defined in this class with the IMSID name (with the IMSCTRL macro) for system access authorization checking at signon.

Database
The database resource class contains a profile for each database defined to RACF for authorization checking. The database group resource class allows grouping of database resources that have a common access authority profile.

Field The field resource class allows RACF authorization checking of fields within a database. The field resource group class lets you group common access fields for RACF authorization checking.

Segment
The segment resource class identifies individual segments to RACF. The segment group resource class permits grouping of segments with a common access authority profile for RACF authorization checking.

APPC/MVS
The APPC resource class identifies transaction profiles for LU 6.2 transactions to RACF.

Other This resource class is installation dependent.

Related Reading: For more information on APPC, see *IMS/ESA Administration Guide: Transaction Manager*.

The names of the transaction, transaction group, and application group name classes are derived from the RCLASS specification in the SECURITY macro. Table 22 shows resource class assignments.

Table 22. Resource Class Assignments

	Resource Class Name	
Resource Class	**RCLASS=IMS**	**RCLASS=xxxxxxx**
Transaction resource class	TIMS	Txxxxxxx
Transaction group resource class	GIMS	Gxxxxxxx
Command resource class	CIMS	Cxxxxxxx
Command group resource class	DIMS	Dxxxxxxx
Database resource class	PIMS	Pxxxxxxx
Database group resource class	QIMS	Qxxxxxxx
Segment resource class	SIMS	Sxxxxxxx
Segment group resource class	UIMS	Uxxxxxxx
Field resource class	FIMS	Fxxxxxxx
Field group resource class	HIMS	Hxxxxxxx
Other resource class	OIMS	Oxxxxxxx
Other group resource class	WIMS	Wxxxxxxx
Application group name resource class	AIMS	Axxxxxxx
Application resource class	APPL	
APPC/IMS	IMS	APPCTPx

The RACF resource classes are defined in the resource class descriptor table (CDT). Initially, the resource classes for which RCLASS=IMS (shown in column 2 of Table 22) are predefined in the CDT. To add the resource class or to define resource classes with user-defined or installation-defined names, you must run the RACF resource class macro, ICHERCDE.

Related Reading: For more information on updating the RACF resource class descriptor table, see *System Programming Library: Resource Access Control Facility (RACF).*

Enabling and Disabling APSB SAF Security

You can enable APSB SAF security using one of the following methods:

- Specify RACF=FULL in the TP scheduler section of the CPI-C application's TP profile and issue the IMS command /SECURE APPC PROFILE. The command /SECURE APPC PROFILE enables APSB SAF Security only for the CPI-C applications that have RACF=FULL specified in the TP profile. The command disables APSB SAF Security for all other CPI-C applications.
- Issue the IMS command /SECURE APPC FULL to enable APSB SAF security for all CPI-C applications.

To disable APSB SAF security, issue the IMS command /SECURE APPC CHECK or /SECURE APPC NONE. With APSB SAF Security disabled, IMS secures the PSB using

Controlling System Startup

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The EXEC parameters for the control region include a way to control the kind of security checking that is done during the current execution. The parameters act as switches for different types of security. They also determine what flexibility the MTO has to override the choice of security checking. You must coordinate the setting of these switches with both overall security design and operational procedures. The parameters are TRN, SGN, RCF, and ISIS.

The values generated for the IMS procedure all specify no security. You must reset them. The choices and parameter values are shown in Table 23 on page 143.

Table 23. JCL Parameters to Control IMS Security

Choice of Security Function	**EXEC Parameter**	**Parameter Value for Security Choice**		
		None	**Yes, with Override**	**Notes**
Transaction authorization	TRN	N	Y, F	1, 2
Signon verification	SGN	N	Y, Z, F, G, M	1, 2, 3
Use RACF (For TRN|SGN)	RCF	N	Y, A, C, S, T	4, 5
Resource access	ISIS	0	1, 2	6
Autosignoff	ASOT	null or 1440	10-1439	7
Autologoff	ALOT	null or 1440	10-1439	b

Table 23. JCL Parameters to Control IMS Security (continued)

Choice of Security Function	EXEC Parameter	Parameter Value for Security Choice		
		None	Yes, with Override	Notes

Notes:

1. With value N, on the /NRESTART command, the MTO can optionally invoke checking.
2. With value Y, the security function is active unless overridden by the MTO.
3. Value M indicates multiple signons for a single user ID. Value Z is equivalent to Y + M; value G is equivalent to F + M.
4. The RACF licensed program is used in conjunction with Command Authorization, Transaction Authorization, or Signon Verification exit routines.
5. If a null value is specified, the choice is the default to that given in system definition.
6. The resource access security checking is required if a non-zero value is given. A value of 1 determines that the RACF exit routine is invoked. A value of 2 is mutually exclusive with a value of 1 and specifies that an installation-written exit routine is invoked.
7. On a terminal defined with ETO, when the last autologon user's last queue is completed, the autologon user immediately signs off without waiting for the autosignoff timeout interval.

You must match the level of the security tables with the suffix identifier for the nucleus. Operational restrictions for the MTO are described in the section "Security Considerations for the Master Terminal" on page 124.

Table 24 shows the JCL parameters and the SECURITY macro parameters they can override.

Table 24. Overriding the Security Macro

JCL Parameter	Security Macro Override
SGN=N	SECLVL=SIGNON SECLVL=FORCSIGN
SGN=Y	SECLVL=NOSIGN
SGN=F	SECLVL=NOSIGN
TRN=N	SECLVL=TRANAUTH SECLVL=FORCSIGN
TRN=Y	SECLVL=NOTRAN
TRN=F	SECLVL=NOTRAN
RCF=N	TYPE=RACFTRM
RCF=Y	TYPE=NORACFTRM
RCF=A	TYPE=NORACFTRM + RACF CMDAUTH (for static terminals)
RCF=T	TYPE=RACFCOM
RCF=C	TYPE=NORACFCM
RCF=S	TYPE=NORACFCM + RACF CMDAUTH (for static terminals)
ISIS=0	TYPE=RACFAGN TYPE=AGNEXIT
ISIS=1	TYPE=NOAGN
ISIS=2	TYPE=NOAGN

If RACF is used for resource access security checking, the dependent region job control can include the USER, GROUP, and PASSWORD specifications on the JOB card. If the user ID specification is omitted, access authority checking is based on the universal access specification for the AGN involved. The AGN name and the IMSID must be included in the EXEC parameter string of the dependent region job control. An example follows:

```
//IMSMPP1 JOB ...USER=MPP1,PASSWORD=PW1
//STEP1   EXEC PGM=DFSRRC00,...,PARM='MSG...,IMS1,AGN1'
```

Implementing Security Changes Online

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

If you plan to use online changes for system definition, the results of the Security Maintenance utility can be made effective for the production environment without restarting IMS. You can:

- Alter transaction and terminal authorization
- Maintain currency of passwords
- Add security provisions to the online system for terminal security, transaction command security, or password security
- Refresh security matrixes that support signon verification or IMS resource access security

First, make your security definitions ready and then execute the Security Maintenance utility. Next, coordinate copies of the inactive IMS.MATRIXA/B and IMS.MODBLKSA/B libraries. Finally, give the MTO instructions to perform an online change using the `/MODIFY` command.

Controlling Security Violations

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Security violations are handled according to the installation's security administration guidelines. IMS records the following security violation attempts on the IMS systemlog:

- Input message from an unauthorized terminal
- Password omitted when one is required
- Password incorrect for authorization
- Misspelled password
- Rejected signon
- Unauthorized DL/I command (`CMD`) call from application program

IMS rejects invalid input messages by sending a message to the terminal entering the message and logging the violation. The IMS system log provides an audit trail for investigation of possible security problems. The IMS system log security violation is identified as a X'10' log record type. You can use the File Select and Formatting Print utility to print the log.

Related Reading: For more information on this utility, see *IMS/ESA Utilities Reference: System*.

Controlling Security Violations

You might want to have tighter security so that you are immediately notified about security violations. You can arrange for the master terminal to be immediately notified about security violations by having messages sent to it whenever the violations occur. To have the master terminal notified when violations occur, specify a non-zero value for the SECURITY macro's SECCNT keyword.

However, in a large network, misspelled passwords, transaction codes, and commands can cause an extremely large number of violations and violation notifications. You can reduce the number of notifications caused by operator errors, while still providing evidence of real attempts to avoid security safeguards, by specifying a notification threshold. When the number of violations from a single terminal equals the notification threshold value (as specified by the SECURITY macro's SECCNT keyword), the master terminal is notified.

Related Reading: For more information on using the SECCNT keyword to set a notification threshold, see "Defining the SECURITY Macro" on page 135.

Another method for recording security violations is available when RACF is installed. Each resource access violation creates a RACF type 80 record. You can use the RACF report writer to create reports based on these records.

Related Reading: For more information on using the RACF report writer to format and print RACF records, see *Resource Access Control Facility (RACF): Auditor's Guide*.

Considering Other Access Control Methods

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	X

This section describes security measures you can take that are not part of the support provided by the Security Maintenance utility. They fall into three general areas:

- "Physical Security"
- "Use of Display Bypass and Password Masking (not DBCTL)"
- "Protecting Your Resources" on page 147

Physical Security

You should consider physical security measures that support your system security. These measures include:

- Controlled access to and from the computer area
- Authorization of DP operations and non-operations personnel in certain terminal areas
- Separately controlled areas for media such as tapes, disks, cards, or files
- Control of computer forms and printed output

Physical security needs are likely to be dynamic and merit periodic review and adjustment

Use of Display Bypass and Password Masking (not DBCTL)

IMS does not provide a software function to blank out or obliterate passwords from the terminal device display media after they are accepted. However, Message

Format Service (MFS) facilities enable users to define fields with a non-display attribute (for 3270 display devices). IMS removes passwords from messages prior to recording them on the log.

If you plan to use passwords as part of transaction and command entry, you should design screen formats to incorporate non-display fields. This protection is especially important for the `/SIGN` command. The DFS3649 signon required message has non-display fields built into it for entering passwords on ACF/VTAM display terminals.

Most key-driven terminals have a feature (called the bypass feature) that permits characters to be entered without displaying them. Ordinarily, a terminal with this feature is operated continuously either in display or bypass mode. If passwords are to be masked to support security requirements, this feature is a necessity.

The bypass feature can be used operationally for establishing standards of protection for not only passwords, but also command verbs, commands, transaction codes, and text.

Protecting Your Resources

You can protect IMS system libraries and data sets, as well as VSAM, OSAM, and Fast Path databases, in both the online (DB/DC, DBCTL, and DCCTL) and batch environments.

IMS system libraries and system data sets

You can use RACF to protect IMS system libraries and system data sets. IMS invokes RACF to determine whether the user ID associated with the system address space (control region, DLISAS, or batch) attempting to open the resource has the necessary access authorization. Actually, when RACF authorizes access, it associates a user ID with the started procedure name (IMS or DLISAS procedure) through a started task table. If you start IMS with JCL, the RACF user ID can be on the job card along with its password.

Related Reading: For more information on this process, see *System Programming Library: Resource Access Control Facility (RACF)*.

If the user ID does not have the authorization, access is denied. The basic rule is “Whoever has the DD card must have the authority.”

IMS procedure: If the IMS procedure is associated with a RACF user ID (with sufficient authority), the IMS control region can open a RACF-protected data set. If an association does not exist, the IMS control region is not allowed to open a RACF-protected data set that does not allow universal access for the requested authority level.

DLISAS procedure: If the DLISAS procedure is associated with a RACF user ID, it overrides the RACF user ID for the IMS procedure. If an association does not exist, the RACF user ID associated with the IMS procedure is used for RACF access checking.

Databases (not DCCTL)

You can protect your VSAM and OSAM full-function databases, as well as your Fast Path DEDBs.

Segment- and field-level sensitivity: Through centralized control over the content of database definitions, program specification blocks, and the libraries in

which they reside, an effective scheme of protection attributes can be assigned to data. Note, however, that for database protection through PSBs to be totally effective, you should also protect the PSB library and the application program library (to safeguard the code that accesses the databases).

Segment-level sensitivity: If you are not using field-level sensitivity, the smallest unit of data that can be protected is the segment. The basic actions that can be authorized are:

None No access to segment type.

Read Segment type can only be retrieved.

One or more of the following additional actions combined with read can be authorized:

Add New occurrences of segment type can be inserted.

Update
An existing occurrence of a segment type can be replaced.

Delete An existing occurrence of a segment type can be deleted.

The way the PCB and the parameter values for the PROCOPT keyword are specified controls the authorization. Although access authorization is declared at the program level, enforcement of the authorization can be made to appear at the transaction code or individual hierarchic level of a database. If only one transaction code is associated with a particular program, then the access authorization is effective at the transaction level. By using SENSEG statements in the PSB and key sensitivity as a processing option for higher-level segments, masking can be effective at the individual hierarchic level.

Field-level sensitivity: As described in *IMS/ESA Administration Guide: Database Manager*, field-level sensitivity can provide another kind of database security. Database descriptions (DBDs) and PSBs can be coded to permit access to a required subset of fields within a segment. Field-level sensitivity can also be used to control the replace function at the field level to help ensure database integrity.

Related Reading:

- For more information on security at the database level, see *IMS/ESA Administration Guide: Database Manager*.
- For information on specifying segment access authorization, see *IMS/ESA Utilities Reference: System*.

RACF security: You use the RACF user ID of the DLISAS or control region started procedure, depending on the environment you are executing. If you start IMS with JCL, the RACF user ID can be on the job card along with its password.

VSAM full-function database: In an online environment, if a RACF user ID is associated with the DLISAS started procedure, that ID is used for access checking. If a RACF user ID is not associated with the DLISAS started procedure, the control region RACF user ID is utilized. (In the batch environment, the user ID of the batch job is employed.) Access authority of “CONTROL” is required. The database must be defined as ICFCATALOG. Use of the older “VSAM” catalog type is restricted.

OSAM full-function database: In an online environment, the RACF user ID of DLISAS is used; in the batch environment, the user ID of the batch job is used.

Fast Path DEDBs: In an online environment, the control region RACF user ID is used. (No batch environment exists.)

Additional protection: You can also implement database security with the DATABASE, FIELD, and SEGMENT classes in RACF.

Related Reading: For more information on these resource classes, see "Preparing to Use RACF for Security" on page 140.

An Alternative to Access Control: Encryption

When preventing access to the data is difficult or impractical, encryption can protect data that is in files or data that is being communicated in a network. IMS offers some file encryption capability (through IMS Segment Edit/Compression exit routines, for example) but no communication encryption capability.

Additional Cryptographic Support

The Programmed Cryptographic Facility, program number 5740-XY5, provides file and communications encryption under MVS. File encryption of the physical hierarchical database keeps unauthorized individuals from looking at the data when the physical disk pack containing the database is removed from its usual area. File encryption support extends to VSAM physical databases. Communications encryption supports ACF/VTAM supported terminals.

Using the Segment Edit/Compression Exit Routine (not DCCTL)

You can use this routine to provide data encryption. By including the IBM Programmed Cryptographic Facility within your exit routine, you can reduce your programming effort. The facility is executed via assembler macro calls. Segments are encrypted before being placed in the database buffer pool. The SEGM control statement in the IMS DBDGEN includes a keyword to specify the name of this exit routine.

Implementing Security Changes Online

If you plan to use online changes for system definition, the results of the Security Maintenance utility can be made effective for the production environment without restarting the IMS. You can:

- Maintain currency of passwords (not DBCTL)
- Add security provisions to the online system for password security (not DBCTL)
- Refresh security matrixes that support IMS resource access security

First make your security definitions ready and then execute the Security Maintenance utility. Next, coordinate copies of the inactive IMS.MATRIXA/B and IMS.MODBLKSA/B libraries. Finally, give the MTO instructions to perform an online change using the `/MODIFY` command.

Controlling Security Violations (not DBCTL)

Security violations are handled according to the installation's security administration guidelines. IMS records the following security violation attempts on the IMS system log:

- Password omitted when one is required
- Password incorrect for authorization

- Misspelled password

Any of these errors causes IMS to log the violation. The IMS system log provides an audit trail for investigation of possible security problems. The IMS system log security violation is identified as a X'10' log record type. You can use the File Select and Formatting Print utility to print the log.

Related Reading: For more information on these utilities, see *IMS/ESA Utilities Reference: System*.

Another method for recording security violations is available when RACF is installed. Each resource access violation creates a RACF type 80 record. For utilities to format and print RACF records, see *System Programming Library: Resource Access Control Facility (RACF)*.

Security Considerations for DBCTL

This section contains the information on establishing security for an IMS DBCTL environment.

Resources That Can Be Protected

Before you decide what security facilities to use in designing a secure IMS system, you should know which resources within the system need protection. In other words, you should decide what to protect before you decide how to protect it.

The following list of resources can be protected:

IMS online system
: The IMS system control program that enables online application programs to process the database through terminals.

PSB
: Program specification block. The control block that describes a group of hierarchic databases and logical message destinations used by an online application program.

BMP application program
: A program in a DBCTL environment that performs work for a user. Batch message programs are activated by the dependent region controller after the region is started by JCL.

Database
: A collection of data that is fundamental to the user's activity. Using a Program Communications Block (PCB), a program has a logical view of the database, as described by the IMS physical database design.

Dependent region
: An area of storage in the IMS online system in which batch or online application programs are executed. The dependent region can be a BMP type region or a CCTL.

System data set
: A collection of data that is fundamental to the operation of the IMS online system. An example is the IMS.MATRIX data set containing initialized security tables.

Security Choices Made during System Definition

You can make IMS security choices in two system definition macros: SECURITY and IMSGEN. [10] Use the SECURITY macro to choose the type of security to be active in online execution. You name the resources using the Security Maintenance utility, alone or with the Resource Access Control Facility (RACF). If you use the SECURITY macro, you do not need to use the IMSGEN macro to define security.

Use the IMSGEN macro to control options of the password and terminal security functions of the Security Maintenance utility. The macro options permit the MTO to override the password and terminal security functions when restarting the IMS system using the /NRESTART command. The MTO should be aware that any security override might compromise resource protection.

Deciding Which Security Facilities to Use

In choosing security facilities, you should consider the information presented in the following sections, as well as your installation's security standards and operating procedures. Table 25 summarizes the resources you can protect and the facilities you can use to protect them.

Table 25. DBCTL Resources and the Facilities to Protect Them

Resources	Security Options/Type of Protection	Facilities
System data set	OS password protection	MVS
	Data set protection (VSAM) (using PERMIT, RDEFINE classes)	RACF
Database	Segment sensitivity	PSBGEN
	Field sensitivity	PSBGEN
	Password security (for /LOCK, /UNLOCK commands)	Security Maintenance
PSB	Resource access security	Security Maintenance and Exit Routine
	Resource access security	Security Maintenance and RACF
BMP application program	Password security (for /IAM, /LOCK, /UNLOCK commands)	Security Maintenance
	Extended resource protection (using APPL keyword)	RACF
Control region	Extended resource protection (using APPL resource class)	RACF
Dependent region	Resource access security	Security Maintenance and Exit Routine
	Resource access security	Security Maintenance and RACF
	Extended resource protection (using AIMS resource class)	RACF

Design Considerations for IMS Security

This section explains how the various choices of IMS security can be used. When you are deciding on each part of your security design, consider the physical actions

10. The Security Maintenance utility and the Resource Access Control Facility (RACF) can also be used to implement security decisions. See *IMS/ESA Utilities Reference: System* for additional information on these utilities.

that an end user must take to obtain access to the system. You will probably use more than one type of security checking. This section assumes:

- A user identification as a control point
- The master terminal as a control point
- The use of RACF protection
- The use of a region as a control point

Using Password Protection with Command Keywords

To provide verification before a command is accepted, you can require an accompanying password. The password is entered within parentheses immediately following the command verb. The password protection provided by the Security Maintenance utility prevents a change of status of database or program resources by the `/LOCK` or `/UNLOCK` command unless a correct password is supplied.

Limiting Access from a Dependent BMP or CCTL Region

Another resource to protect is the dependent region. You can protect this resource by preventing the start of an unauthorized dependent region by start of task JCL and by preventing the use of unauthorized resources in a dependent region. Resource access security provides this protection.

You can authorize PSBs to be used by BMP and CCTL regions. In order to control what programs are authorized to use a dependent region, you group them under an application group name (AGN). The JCL that starts up a BMP region stipulates which group of application programs are eligible to be scheduled, if the AGN is known to the control region. For a CCTL, the AGN parameter comes from the DRA startup table. Further validation restricts that region's use to PSBs associated with the AGN.

The use of the AGN ties the two-part protection together. The dependent region and resource authorization is implemented by the Security Maintenance utility and an exit routine or by the Security Maintenance utility and RACF. You can prevent an unauthorized BMP or CCTL from connecting to DBCTL by performing the following four steps:

1. Code the AGN parameter. The AGN parameter name associated with the JCL is the same name associated with the profile of the limited resources permitted to use the dependent region.
2. Use the Security Maintenance utility to create an entry in the AGN table, with the same name used by the AGN parameter. Then you define the resources permitted to use the dependent region. This step alone provides the second part of resource access security. A different AGN name must be assigned to every BMP or CCTL region.
3. Use the Security Maintenance utility and an exit routine to authorize the region. The exit routine must compare the AGN entries in the application group name table to the AGN name associated with the region. A mismatch prevents starting the BMP region or the connection of a CCTL to the DBCTL environment. For the Security Maintenance utility and RACF, a different user ID is assigned to the job cards of all dependent region JCL. These same user IDs are entered into the RACF system.
4. Create an entry in the class descriptor table (CDT) for the AGN resource for RACF. If you choose not to take the default description, the ICHERCDE macro creates the entry in the CDT. All AGN names assigned to dependent region JCL are logically related to the CDT resource entry by the RACF RDEFINE

statement. RACF connects the user ID to its appropriate AGN source. RACF prevents starting a dependent region if the user ID of the start-of-task JCL is not permitted.

The ISIS parameter on the IMS control region EXEC statement controls the use of resource access security. If a value of 2 is specified, an exit routine checks the request to use the dependent region. If a value of 1 is specified, RACF is used instead of an exit routine.

Associating Resources with Application Group Names: To define the resources authorized for dependent regions, you must name the application groups and their authorized PSBs. You can define up to 5000 AGNs by means of resource access statements as shown in "Examples of Security Maintenance Utility Input Statements" on page 154. There is no limit to the number of PSBs that can be defined per AGN.

If a PSB is selected to be scheduled in a dependent region, that PSB must have been declared through the Security Maintenance utility as valid for the associated AGN. Otherwise, the scheduling of that PSB is rejected.

Designing a Resource Access Exit Routine: The Resource Access exit routine (DFSISIS0) authorizes starting a dependent region. A region starts when a match occurs between a name entry in the application group name table and the name assigned to the AGN keyword of the EXEC statement in the BMP JCL or the CCTL DRA startup table. When the IMS control program calls this exit routine, two registers contain pointers to the sources that are to be compared by this exit routine. The DFSISIS0 routine supplied with IMS must be replaced. (The routine supplied refuses authorization to all callers by notifying the IMS control program that an invalid dependent region has tried to start.) The use of the exit routine is specified by coding the AGNEXIT parameter for the TYPE keyword in the SECURITY macro.

Related Reading: For more information on register usage, see *IMS/ESA Customization Guide*.

Limiting Access from a CCTL

You can control resource access and prevent an unauthorized CCTL from connecting to the DBCTL environment using the ISIS execution parameter. If you specify ISIS = 1 or ISIS = 2, both the CCTL connection and PSB scheduling is checked. If you specify ISIS = 0, neither is done.

- **Select ISIS = 1**. Build RACF tables that define valid user ID-AGN combinations. The JOB statement for a CCTL to be run contains the user ID; the DRA startup table for the CCTL contains the AGN. If these do not correspond to an entry in RACF's tables, the CCTL cannot connect.
- **Select ISIS = 2**. Create a Resource Access Security exit routine, named DFSISIS0. (See "Designing a Resource Access Exit Routine" for an explanation of how to specify that you use it.) Your routine must determine whether the AGN passed to it is valid for the attempted connection.

  Related Reading: For more information on register usage, see *IMS/ESA Customization Guide*.

For PSB-scheduling protection, the PSB must be an AGN defined by SMU, and the CCTL must have this AGN in its DRA startup table.

Security Considerations for Fast Path Application Programs

When designing security protection for Fast Path application programs, or for DL/I programs that access Fast Path databases, consider that the processing in a dependent region can be protected by resource access security. You assign an AGN to the region and authorize a PSB.

Activating IMS Security

This section gives guidance on the steps you take to activate your IMS security design, using the Security Maintenance utility, RACF, and program exit routines. Depending upon the security facilities you choose to use, you must perform the tasks as explained in the following sections:

- "Defining the SECURITY Macro"
- "Preparing to Use the IMS Security Maintenance Utility"
- "Preparing to Use RACF for Security" on page 156

Defining the SECURITY Macro

The purposes of the macro keywords are given here.

Related Reading: For more information on specifying macro keywords, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

TYPE Keyword: Using this keyword, you specify which security facility, or combination of facilities, is to be included in your system. The parameter to invoke security checking is:

AGNEXIT
: For resource access security. An installation-written exit routine checks which programs are authorized under the AGN to use the region. Further validation restricts that region's use to PSBs, transaction codes, or LTERM names associated with the AGN.

If you plan to use the RACF licensed program for your MVS system, specify:

RACFAGN For use of a RACF exit routine to verify AGN validity. The AGN names are defined as a protection group to RACF.

RCLASS Keyword: The RCLASS keyword specifies an identifier to be used to identify the IMS system as a resource class to RACF for AGN verification. Resource class assignment is described in Table 26 on page 157.

PASSWD Keyword: This keyword controls the options available to the DBCTL operator. You can use these parameters to control the amount of security-checking flexibility given the operator for the current online execution.

Preparing to Use the IMS Security Maintenance Utility

The first step toward activating the Security Maintenance utility is to provide input control and data statements. A control statement names the resource to be protected and the data statement names the security to be established for the named resource.

Related Reading: For information on the rules for coding the Security Maintenance utility, see *IMS/ESA Utilities Reference: System*.

Examples of Security Maintenance Utility Input Statements:

- Passwords assigned to each BMP program

```
)( PROGRAM     ACCT
    PASSWORD   DOLLAR

)( PROGRAM     ENG560
    PASSWORD   PARTNO

)( PROGRAM     LOGREC
    PASSWORD   NONE

)( PROGRAM     AGC0568
    PASSWORD   MONEY
```

- Passwords assigned to each database

```
)( DATABASE    ACCTLOG
    PASSWORD   LOG

)( DATABASE    ACCTREC
    PASSWORD   REC

)( DATABASE    PARTSREC
    PASSWORD   PIERSQ

)( DATABASE    PARTSREC
    PASSWORD   ASSY
```

- Resource access security for several regions

```
)(  AGN   TEST001
    AGPSB DDLTBP01
)(  AGN   TEST002
    AGPSB ALL
)(  AGN   TEST003
    AGPSB APOL1
    AGPSB A3270
    AGPSB GISBMP09
)(  AGN   TEST004
    AGPSB A3270
)(  AGN   TEST005
    AGPSB INTCON
```

Executing the IMS Security Maintenance Utility: The Security Maintenance utility is run after completion of each system definition because the internal system description blocks, created by the system definition process, are used as input to the Security Maintenance utility. The utility is run for every modification of the existing resource access profiles and when a new version of the security tables in IMS.MATRIX is required. The SECURITY member in IMS.PROCLIB contains the required JCL to read the input Security Maintenance utility control and data statements and to create the security tables in IMS.MATRIX. You can modify the procedure JCL to match the input data device type. Use MVS password or RACF data set protection to protect IMS.PROCLIB and the data set containing the Security Maintenance utility control and data statements.

The Security Maintenance utility runs as a three-step job. The first step accepts the input control and data statements and checks them against the IMS system being maintained to ensure correct format and validity. When no errors are detected in the first step, the second step, an operating system assembly, is performed. Step 3 is a link-edit that takes the assembly output from step 2 and creates the following:

- Password offset list (DFSISPLx)
- Application group name table (DFSAGT0x)

Depending on the input presented, a variable number of output load modules are created as members of the IMS.MATRIX data set. These members cannot be reprocessed using the linkage editor.

Allocating the IMS.MATRIX Data Set: The maximum size of any generated matrix in bytes (M) is:

```
M = (I X R)/8
```

where:

I The total number of unique passwords in the Security Maintenance utility input.

R The total number of resources associated with a password.

Controlling Versions of the Security Matrix Tables: Additional versions of the Security Maintenance utility security tables are created with each run of the Security procedure. Eight security table members in IMS.MATRIX carry a suffix version code.

The suffix version codes are controlled by the IMS symbolic parameter in the EXEC statement. This alphanumeric parameter must match the alphanumeric suffix of the IMS nucleus used in the next IMS restart. The OPTN symbolic parameter in the PARM keyword of the Security Maintenance utility EXEC statement controls the update of the current security tables.

To verify the resource entries before updating the security tables, specify LIST as a parameter value to check validity and list the new security tables. The UPDATE parameter checks the validity of, lists, and updates the security tables in IMS.MATRIX. After the update occurs, this version is not used until the next IMS restart or until an online change activates it. The last version of security tables is resident in main storage as a result of the previous system restart. On every Security Maintenance utility update run, the last security tables created in IMS.MATRIX are overwritten by the current version of the security tables. To save more than one version of the security tables, you must add JCL to the security procedure. Alternatively, when IMS.MATRIX is used as a staging library, you can copy it using the Online Change utility.

Preparing to Use RACF for Security

You can implement a security plan using RACF by performing these steps:

1. Prepare a list of all the IMS online resources to be protected, arranging them in groups to give an overview of the total resources covered.
2. Select the security facilities that protect the resource groups.
3. Design and list resource access profiles.
4. Code the SECURITY and IMSGEN macros.
5. Code the resource access profiles, using the Security Maintenance utility.
6. Describe the system resource classes to RACF.
7. Add users, groups, and data sets to RACF.
8. Execute the Security Maintenance utility.
9. Modify JCL procedures in IMS.PROCLIB.

RACF resource classes are used by the IMS security function. The application group resource class holds a profile for every AGN defined to RACF. The name of the resource class is derived from the RCLASS specification in the SECURITY macro. Table 26 on page 157 shows resource class assignments for DBCTL.

Table 26. Resource Class Assignments for DBCTL

	Resource Class Name	
Resource Class	**RCLASS = IMS**	**RCLASS = xxxxxxx**
Application group name resource class	AIMS	Axxxxxxx

The RACF resource classes are defined in the resource class descriptor table (CDT). Initially, the AIMS resource class is predefined in the CDT. To add the resource class or to define resource classes with user-defined names, you must run the RACF resource class macro, ICHERCDE.

Controlling System Startup

The EXEC parameter ISIS for the control region includes a way to control the kind of security checking that is done during the current execution. The parameter determines what flexibility the DBCTL operator must override the choice of security checking. You must coordinate the setting of the ISIS parameter with both overall security design and operational procedures.

The values generated for the DBC procedure all specify no security. You must reset them. The choices and parameter values are shown in Table 27.

Table 27. JCL Parameters to Control IMS Security

		Parameter Value for Security Choice		
Choice of Security Function	**EXEC Parameter**	**None**	**Yes, with Override**	**Notes**
Resource access	ISIS	0	1, 2	1

Notes:

1. The resource access security checking is required if a non-zero value is given. A value of 1 determines that the RACF exit routine is invoked. A value of 2 is mutually exclusive with a value of 1 and specifies that an installation-written exit routine is invoked.

You must match the level of the security tables with the suffix identifier for the nucleus.

Table 28 shows the JCL parameters and the SECURITY macro parameters they can override.

Table 28. Overriding the Security Macro in a DBCTL Environment

JCL Parameter and Value	Security Macro Override
ISIS = 0	TYPE = RACFAGN TYPE = AGNEXIT
ISIS = 1	TYPE = NOAGN
ISIS = 2	TYPE = NOAGN

If RACF is used for resource access security checking, the BMP region job control can include the USER, GROUP, and PASSWORD specifications on the JOB card. If the user ID specification is omitted, access authority checking is based on the universal access specification for the AGN involved. The AGN name and the IMSID must be included in the EXEC parameter string of the dependent region job control. An example follows:

```
//IMSMPP1 JOB ...USER=MPP1,PASSWORD=PW1
```

Security Considerations for DBCTL

If RACF is to be used for a CCTL, the CCTL JOB card must include the USER specification. The CCTL itself can provide security checking (RACF or its own) independent of the DBCTL security checking described in this chapter.

Implementing Security Changes Online

If you plan to use online changes for system definition, the results of the Security Maintenance utility can be made effective for the production environment without restarting the IMS. You can:

- Maintain currency of passwords
- Add security provisions to the online system for password security
- Refresh security matrixes that support IMS resource access security

First, make your security definitions ready and then execute the Security Maintenance utility. Next, coordinate copies of the inactive IMS.MATRIXA/B and IMS.MODBLKSA/B libraries. Finally, give the MTO instructions to perform an online change using the `/MODIFY` command.

Controlling Security Violations

Security violations are handled according to the installation's security administration guidelines. IMS records the following security violation attempts on the IMS system log:

- Password omitted when one is required
- Password incorrect for authorization
- Misspelled password

Any of these errors causes IMS to log the violation. The IMS system log provides an audit trail for investigation of possible security problems. The IMS system log security violation is identified as a X'10' log record type. You can use the File Select and Formatting Print utility to print the log.

Another method for recording security violations is available when RACF is installed. Each resource access violation creates a RACF type 80 record.

Related Reading:

- For more information on the File Select and Formatting Print utility, see *IMS/ESA Utilities Reference: System*.
- For more information on utilities to format and print RACF records, see *System Programming Library: Resource Access Control Facility (RACF).*

Chapter 5. Designing a System with Extended Recovery Facility

An Extended Recovery Facility (XRF) complex allows you to maintain continuity of online transaction processing by giving you the ability to rapidly resume end-user service when a failure occurs.

In this Chapter:

The XRF is a set of functions within existing programs that can be used by an installation to achieve a high level of availability for selected end users. In particular, it responds to the IMS users' need for continuity of online transaction processing. The required set of IBM software products that work together with IMS to make up XRF are:

- MVS/ESA
- DFSMS
- VTAM
- NCP
- SSP

They run on any IBM processor that can operate in System/390 extended architecture mode to provide your end users with improved reliability and continuity of service.

The XRF approach to high availability builds on two assumptions:

- Many IMS installations try to minimize both planned and unplanned outages. These installations are willing to devote extra resources to improve the service for their high-priority work.
- A defect that causes a failure in one environment does not necessarily cause a failure in a different environment.

XRF does not eliminate outages. Even if all unplanned outages (hardware and software failures) disappeared, planned outages (maintenance, configuration changes, and migration) would still occur. Although XRF does not provide continuous availability, it does reduce the impact of planned and unplanned outages on end users.

Related Reading: The remote site recovery (RSR) feature provides many of the same benefits that XRF provides. For a comparison of XRF and RSR, see Table 41 on page 356.

What Kind of Installation Can Benefit from XRF?

Among the data processing installations that are especially sensitive to disruptions in service are banking institutions, credit verification companies, and manufacturing-process control centers. In these and other installations, loss of service can cause end-user dissatisfaction and loss of business.

Because of the growth of interactive workload and critical business applications, availability has become a major concern for many IMS installations. An installation that uses IMS to log employees on and off their jobs during shift change, for example, needs high availability during two brief periods of each work day. A bank that uses IMS to process transactions needs high availability from 6 a.m. to 10 p.m. An installation that requires uninterrupted processing 24 hours a day, 7 days a week, regards outages of 1 to 2 hours as unacceptable. Although XRF cannot eliminate outages at these installations, it can remove much of the disruption to end users.

Generally, an installation that can benefit from XRF has:

- Multiple large-scale systems in a single operational environment
- Many end users served by IMS
- A requirement for very high availability to end users

Such an installation typically operates in a well-structured and tightly controlled environment, one where the effective installation management needed to implement XRF exists.

Concepts and Terminology

High availability with only short interruptions in end-user service can be difficult to achieve for IMS failures. Before restart can take place, termination processing might need to be performed and the cause of failure determined. Restart itself can be time consuming if regions must be restarted and terminal sessions reestablished.

An XRF complex allows you to rapidly resume of end-user service when a failure occurs. The configuration consists of a primary subsystem (called the active IMS), usually operating in a preferred processor, and an alternate subsystem (called the alternate IMS), tracking the activities of the active IMS concurrently in the same or in a different processor.

The active IMS processes the IMS workload in the same way that it would without XRF. It does not do any more work than its normal processing and logging. Through the log, it informs the alternate IMS of its activity. A combination of surveillance mechanisms (the log, RDS, and an ISC link), alert the alternate IMS to problems in the active IMS. The active IMS is not aware of the activities of the alternate IMS.

The alternate IMS does not process any of the IMS transactions that are entered in the active IMS; it monitors the active IMS for an indication of trouble. The alternate IMS records resource changes in the active IMS and updates its control blocks and buffers, continuously changing its status to match that of the active IMS. This alternate IMS is in a state of readiness so that work can quickly shift from the active IMS to the alternate IMS without waiting for dependent regions to be started, data sets to be allocated and opened, and sessions to be established. This shift of the workload from the active IMS to the alternate IMS is called a *takeover*. After a takeover has occurred, you can establish a new alternate IMS.

When an IMS failure occurs, or when failures that affect the MVS operating system or entire processor occur, the alternate IMS assumes the workload of the active IMS. If end users are using appropriate terminals, they experience little disruption when a takeover occurs. The effect is of a single system image; the end user need not know which IMS subsystem is being used and might even be unaware that a switch in the IMS host system occurred.

A takeover can occur when the active IMS abends. It can also occur for some of the other common causes of outages at an IMS installation, such as:

- A surveillance-detectable IMS failure
- A surveillance-detectable MVS failure, loop, or wait state
- A central processor complex (CPC) failure
- A VTAM failure that results in a TPEND exit
- An IRLM failure that results in a STATUS exit

You can also invoke XRF to introduce planned changes into a production environment with minimal disruption to end users.

One of the major service elements in this chain is the processor. In this chapter, the acronym CPC—central processor complex—replaces the word “processor”. A CPC is a processor running under the control of a single OS/390 operating system. It can be either a uniprocessor or a multiprocessor (including a dyadic processor).

Where XRF requires a 3705, 3725, or 3745 Communication Controller (CCU), the CCU is referred to as 37x5.

XRF Systems

The operators of both IMS subsystems in the complex must enlarge their area of concern and responsibility to include the other system in the XRF complex. Whenever either IMS subsystem changes status, both operators should know about it.

Both operators must be aware of the relationship between elements in the XRF complex:

- The active IMS subsystem processes the IMS workload.
- The alternate IMS subsystem maintains a state of readiness to take over the workload of the active IMS.
- The active and alternate CPCs can both have non-IMS work running on them.
- The active and alternate subsystems work as a unit to form a recoverable service element (RSE). The two IMS subsystems in an RSE share an RSENAME and a VTAM USERVAR. You can start either subsystem as the active IMS (subject to operational restrictions), then start the other as the alternate IMS. The alternate IMS might initiate a takeover if it detects a failure on the active IMS. Both identify themselves to the availability manager component of OS/390.
- An element of the XRF complex that has an alternate IMS subsystem but is unable to initiate a takeover is a dependent service element (DSE). The CPC, OS/390, VTAM, and the IRLM are DSEs. A DSE depends on IMS to recognize when it fails and to request a takeover for it.
- A shift of workload from the active to the alternate IMS is a takeover. Two kinds of takeovers are:

  - A planned takeover, which occurs when you request it because you want the IMS workload shifted from the active IMS to apply hardware or software maintenance.
  - An unplanned takeover, which occurs when the active CPC, OS/390, IMS, or VTAM fails.
- The 37x5 and the DASD service elements are neither recoverable nor dependent; they are shared by both the active and the alternate IMS. A major failure in either one might terminate service to end users.

DBCTL Capabilities

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

A DBCTL environment alone does not have full XRF capability. It does, however, allow you to have a standby, alternate DBCTL environment. This alternate system does not track the active system. You can only bring it to a point where it is fully initialized and ready to take over. The alternate system is then called a pre-initialized DBCTL environment. If the active system fails, you can send a restart command to the alternate system.

You can run a DB/DC environment that provides DBCTL service to a CCTL. If that DB/DC environment runs XRF, the DBCTL service provided is fully XRF capable. Either the operator or the CCTL can issue a switch command to start the takeover.

Related Reading: For more information on requirements and setup of DBCTL service, see *IMS/ESA Operations Guide*.

End-User Service

The user's perception of the service a computer gives is simple—either it works or it does not. However, the system programmer, who is responsible for keeping the computer's service working properly, sees this service as a chain of discrete services—or service elements. And a failure of any one service element can interfere with the user's service.

The elements that are essential to the terminal user's service are like the links in a chain. If one of these links breaks (that is, if one of these service elements fails), the user does not have service.

The links representing service elements could resemble the section of chain as shown in Figure 10.

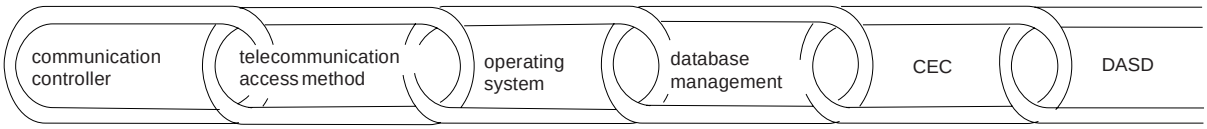


Figure 10. The Service Element Chain

Other hardware and software elements are also in the chain, such as terminals, network programs, channel-to-channel adapters, and communication lines.

Some of the specific service elements in a typical data processing installation are shown in Figure 11.

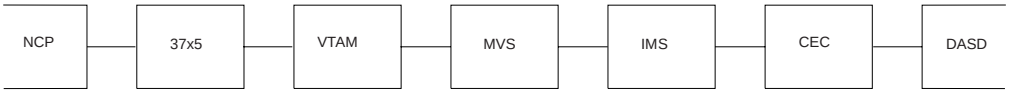


Figure 11. Typical Service Elements

One way to improve availability is to provide standby high-level services, such as systems and CPCs. XRF provides a standby functional environment for IMS. Figure 12 shows that the service element IMS and its supporting environment (MVS, VTAM, and the CPC) are duplicated. If a service element such as a DASD that contains a shared critical data set or an IBM 37x5 Communication Controller fails, service to the terminal user is disrupted; these are single points of failure. But if IMS, VTAM, MVS, or the CPC fails, XRF provides an alternate path. Because of the duplicate elements, these service elements are recoverable; service to the end user resumes quickly.

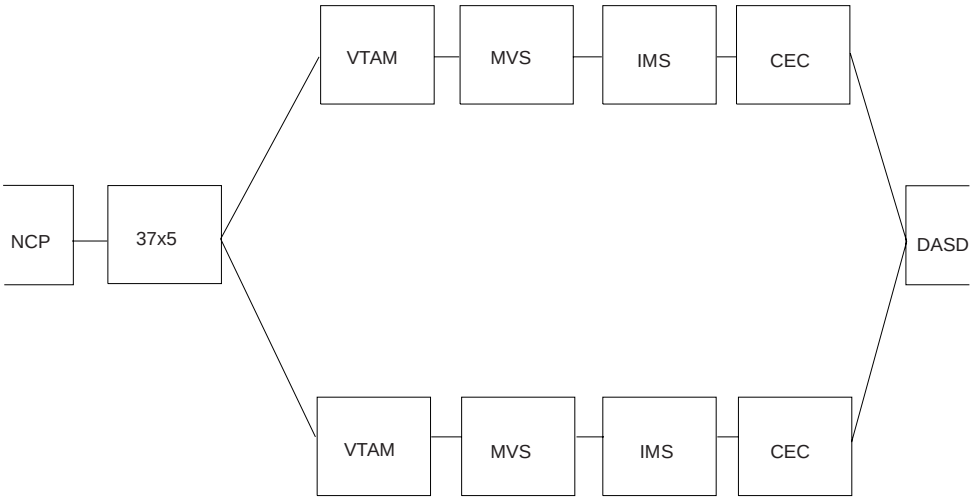


Figure 12. The XRF Concept of End-User Service

What Is an XRF Complex?

To use XRF in your installation, you must build a suitable configuration, called an XRF complex. An XRF complex consists of all the hardware (such as the CPCs and DASDs) and the licensed programs that reside on the active and alternate subsystems to provide XRF for your IMS system. Figure 13 on page 164 shows the typical set of resources in an XRF complex:

- Two CPCs capable of operating independently, such as two IBM ES/9000 CPCs
- MVS (including DFSMS), IMS, and VTAM running in each CPC
- IMS system logs—the online data sets (OLDSs) and write-ahead data sets (WADSs)—shared by the active and the alternate IMS subsystems
- Databases with access paths from the active and the alternate IMS subsystems
- NCP running in each IBM 37x5 Communication Controller that controls the SNA terminals at boundary nodes
- Remote terminals that can communicate with either of the IMS subsystems

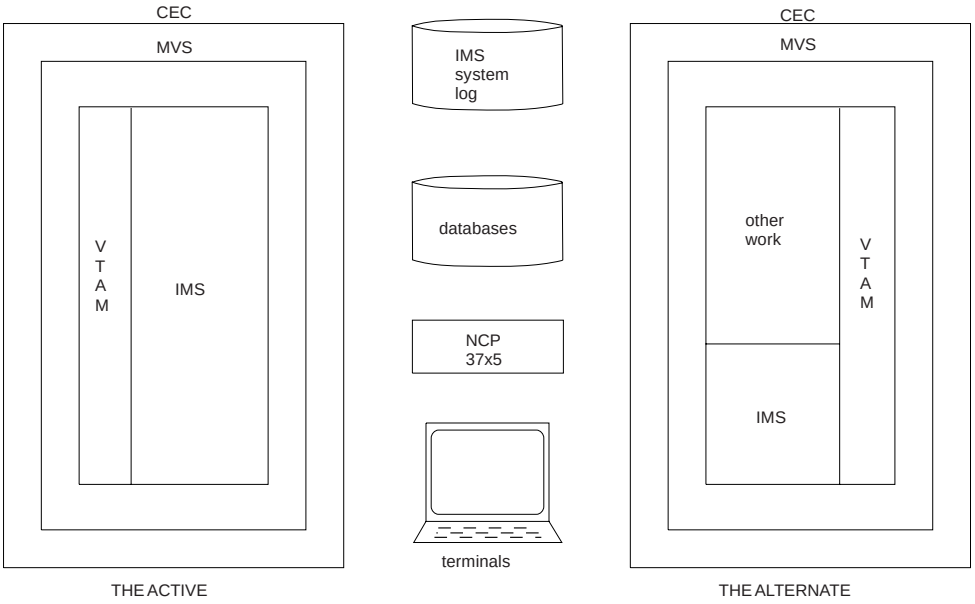


Figure 13. An XRF Complex

The active subsystem processes the high-priority IMS workload in the same way that it would without XRF, except that IRLM is not required for XRF. The alternate subsystem does not process any IMS transactions; it monitors the active subsystem for an indication of trouble. The tasks of the alternate subsystem are to track the progress of the active subsystem and to update its own control blocks so that work can shift quickly from the active to the alternate IMS.

XRF provides a terminal user at your installation with one of three types of service depending on the kind of terminal, the teleprocessing access method that controls it, how it is connected to the alternate subsystem, and how your system programmer defines the terminals. For an SNA terminal that is connected through an NCP and a VTAM that support XRF, the takeover does not terminate a session with XRF IMS. These terminals are called Class 1 terminals. For all other terminals, a takeover disrupts sessions. For some terminals, such as those controlled by BTAM, IMS immediately tries to reestablish service at a takeover. These terminals are Class 2 terminals. For other terminals, such as those terminals that the operator must manually switch to the alternate subsystem at takeover, no XRF support is available at takeover. These are Class 3 terminals. (All LU 6.2 devices are Class 3.) Users at these terminals depend on an operator to reestablish a session with XRF IMS.

Related Reading: For more information on terminal classes, see “Terminals in an XRF Complex” on page 202.

All program temporary fixes (PTFs) that must be applied are identified. Both systems must be at the same IMS release. A planned takeover cannot be used to migrate from one IMS release to another.

Although the alternate subsystem has XRF-related tasks to perform, the CPC, MVS, and VTAM in the alternate subsystem are available to process other work as well. Plan the workload carefully, keeping in mind that, at takeover, service is degraded for the non-XRF work. For example, if you run TSO in the alternate subsystem, MVS might swap out the TSO user address spaces at takeover.

What Is a Takeover?

The best way to describe a takeover is through an example. Consider a typical installation with a need for high availability. The installation in this example is a manufacturing control center that processes production data coming in from many remote terminals. Processing includes updating the IMS databases and recording activity on the IMS system log. Accompanying tasks are running payroll, producing a company report, and performing routine maintenance, tasks that are often lower in priority. This installation runs MVS with VTAM as the teleprocessing access method.

The failure in this example is an IMS control region abend. The following sections describe what might happen at this installation if XRF is not installed and then what happens with XRF installed.

Without XRF

In an installation with one system and without XRF, the IMS control region abend disrupts all service at the terminals. If the system is unavailable long enough, production lines shut down. The operator, the system programmer, or both might have to:

1. Wait for a diagnostic dump
2. Determine the cause of the abend
3. Restart IMS
4. Restart the associated dependent regions
5. Reestablish sessions with terminals

These activities can take from one hour to several hours depending on the type of outage and the size of the installation. Another half-hour might be required to recover IMS to where it was before the outage. Although the original problem might have been a minor one, the recovery is costly. End users can be without service for almost two hours while the operational staff performs the recovery steps. Regaining processing ability and ensuring the integrity of the databases and message queues during the down period is necessary for this installation.

With XRF

An installation with XRF installed is physically different from the installation just described. Figure 14 on page 166 shows the two systems at the XRF complex. Each system includes the CPC and three licensed programs: MVS (including DFP), IMS, and VTAM.

Takeover

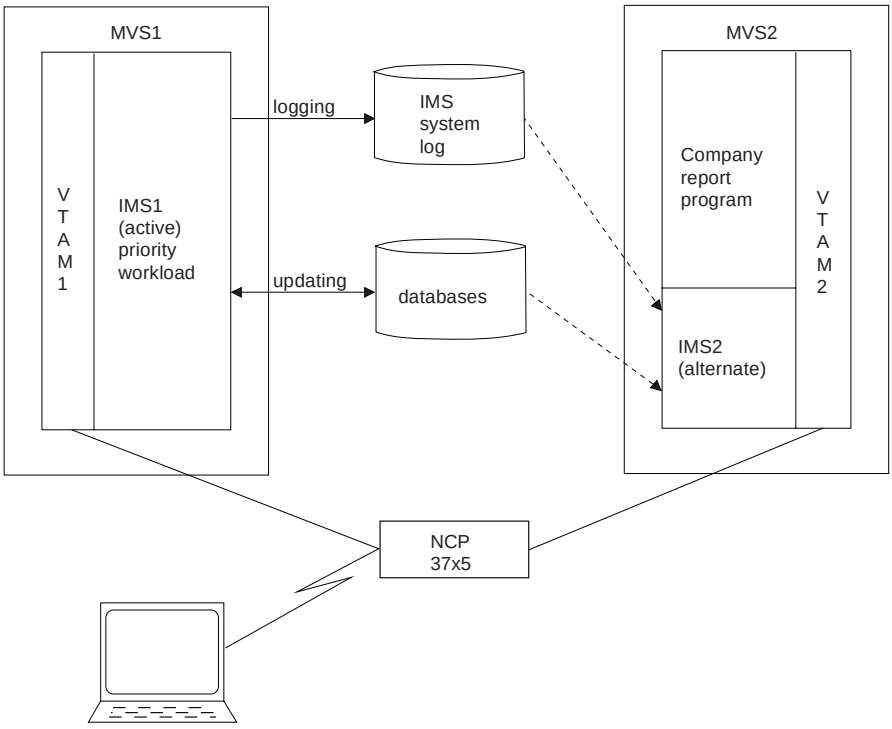


Figure 14. The XRF Complex before a Takeover

IMS1 processes the high-priority work—production data that comes in from the remote terminals. It updates the databases and records its activity on the IMS system log.

IMS2 tracks the active subsystem by monitoring the records on the IMS system log. It also opens backup sessions for Class 1 terminal users that log onto the active subsystem. To maintain an environment identical to that in the active subsystem, IMS2 updates many control blocks and message queues in the alternate subsystem to reflect those in the active subsystem. CPC capacity and storage not used by this activity support the company report program.

When IMS1 abends, the takeover begins. Depending on XRF's demand for real storage, MVS2 might swap out the company report program. IMS2 shifts the production workload to the alternate subsystem and begins to serve Class 1 and Class 2 terminals. Problem determination activities can begin on the failing IMS1.

While IMS2 recovers data and tells NCP to switch sessions on Class 1 terminals, IMS2 and MVS1 prevent IMS1 from writing to the IMS system log and the databases. IMS2 isolates the log and proceeds with the takeover. At the same time, MVS1 performs I/O prevention; it makes sure that all new I/O requests to the databases from IMS1 return without being executed. When MVS has completed or canceled all existing I/O requests to the database data sets, it notifies the operator. Figure 15 on page 167 shows the two systems during the takeover.

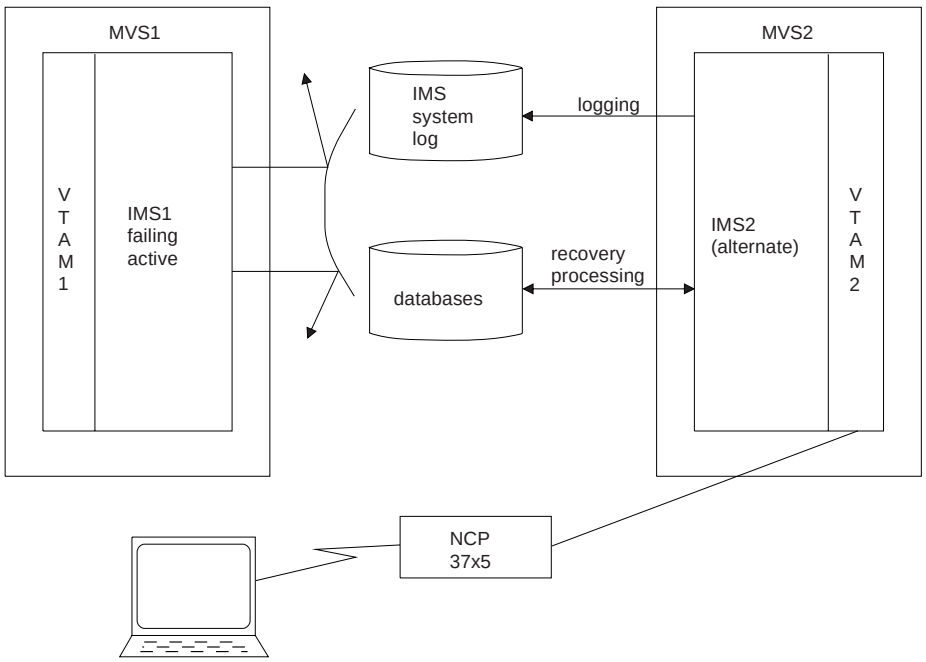


Figure 15. The XRF Complex during a Takeover

The takeover is complete when all the users at Class 1 terminals can communicate with IMS2 and can enter transactions and receive replies from their IMS applications. When IMS2 learns that the failing IMS1 cannot write to the databases, it stops protecting them. Figure 16 shows the XRF complex after the takeover. XRF does not perform problem determination for the installation. It does, however,

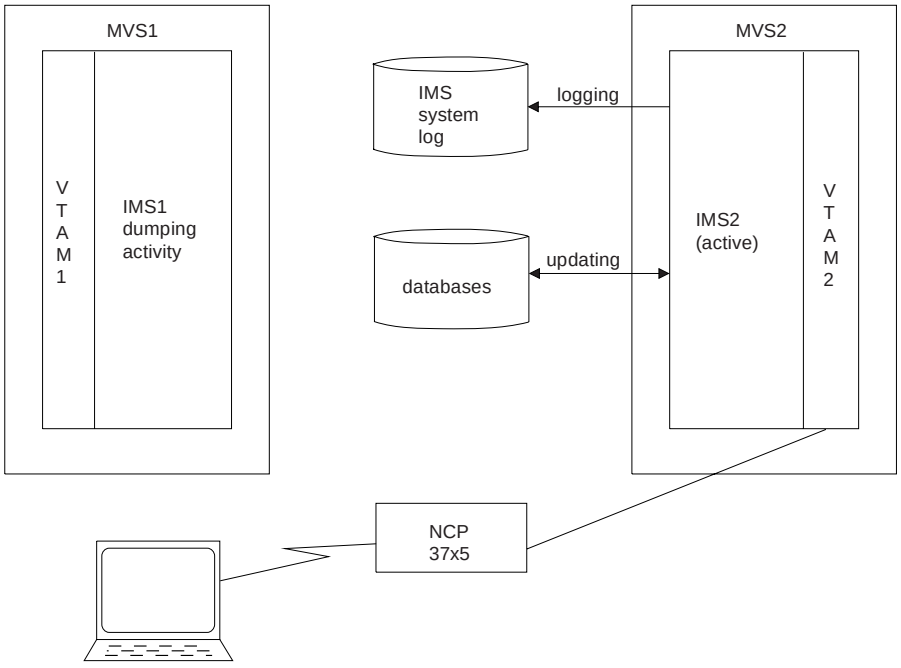


Figure 16. The XRF Complex after a Takeover

reduce the pressure that exists in a non-XRF installation while the terminal users are without service during dumping and restarting. The installation personnel still must determine the cause of the problem in the failed active subsystem. When

dumping activity completes on the failed active subsystem, an operator can return MVS1, IMS1, VTAM1, and CEC1 to service as the alternate subsystem for MVS2, IMS2, VTAM2, and CEC2. If the dumping activity is lengthy, the operator might bring up a third system as an alternate. Until an alternate subsystem exists for the system that is processing the requests of the IMS users, no XRF complex exists.

During the takeover, IMS and MVS send messages to their system operators to notify them of the progress of the takeover. Depending on the cause of the takeover and the resources at the installation, the operators have some tasks to perform.

Related Reading:

- For information on how MVS, IMS, VTAM, and NCP contribute to the XRF takeover process, see "How Does Each Licensed Program Contribute to the XRF Process?" on page 170.
- For information on the messages that XRF sends to the operators at takeover and the appropriate operator responses, see *IMS/ESA Operations Guide*.

Takeover Conditions

A takeover condition is an event that causes IMS in the alternate system to request a takeover. In an installation that runs XRF, an IMS control region abend is always a takeover condition. Parameters that the system programmers specify when they tailor the IMS subsystem determine whether any of the following failures are also takeover conditions:

- An MVS failure, loop, or wait state
- A CPC failure
- A VTAM failure
- An Internal Resource Lock Manager (IRLM) failure

Not all failures at an IMS installation qualify as takeover situations. XRF does not address the outages caused by failures of service elements you do not duplicate. For instance, XRF does not respond to:

- A channel or link failure that causes a break in communication between the CPC and the communication controllers or DASDs
- Failures in the telecommunication network, such as communication controllers, NCP, lines, and terminals
- Intersystem failures, such as those caused by JES3 or CTCs
- Loss of or damage to the IMS databases
- A power failure that affects both CPCs in the complex
- Failures of user catalogs that point to data sets, such as databases

Each installation must make some planning decisions about the takeover. You decide some of the conditions that initiate a takeover, the amount of operator involvement, which terminal sessions recover automatically at takeover, and which methods the alternate system uses to learn of problems in the active system.

Planned Takeover

Your installation can take advantage of a takeover to schedule certain kinds of changes to the two systems. While a system is an alternate, your operator can bring it down, leaving the active system without backup support for the short time it takes your system programmer to perform the updates. The operator then restarts the alternate system with the code or hardware maintenance in place or IPLs MVS

with configuration changes in place. To make the changes on the active system, the operator initiates a takeover and repeats the process on the former active system.

What Does XRF Require?

Building an XRF complex requires additional processing, storage, and communication resources. How many additional resources depends on:

- What resources your installation already has
- How heavily utilized your CPCs are

This section describes the software, hardware, and operational requirements for XRF.

XRF Licensed Program Requirements

To use XRF, IMS must have the same system definition in the active and the alternate systems. Before XRF can be operational, both the active and alternate systems must have the required levels of OS/390, DFSMS, NCP, SSP, and VTAM installed (although these licensed programs do not need to be installed at the same time).

Related Reading: For information on the required levels for these licensed programs, see *IMS/ESA Release Planning Guide*.

Recommendation: The use of Network Communications Control Facility (NCCF) Version 2 Release 2 is recommended, though not required.

XRF Hardware Requirements

XRF works with the following IBM hardware products:

- CPCs

  XRF runs on CPCs supported by OS/390. The CPCs can be different models.
- Communication controllers

  To automatically switch sessions at takeover, XRF requires that you connect the Class 1 terminals, Class 2 terminals, and intermediate routing nodes (IRNs) to 37x5 Communication Controllers.
- Terminals

  All the terminals in your installation that presently use IMS can operate in an XRF complex. The level of support varies, as described in “What Is an XRF Complex?” on page 163.

X·25 Terminal Requirements for XRF Support

Terminals using the X.25 communications protocol can be used with IMS. X.25 terminals can have Class 1, 2, or 3 XRF support, based on their definition in IMS and their X.25 characteristics. X.25 NCP (Network Control Program) Packet Sw Version 2 Release 4 participates in XRF terminal switching for eligible SNA terminals that are in session with IMS. Eligible SNA terminals attach to X.25 networks with the following:

- Integrated X.25 adapters
- Network Interface Adapter (5973-L02)
- External SDLC/HDLC PAD

Only eligible SNA terminals can receive Class 1 XRF support.

Related Reading: For more information on attaching X.25 terminals, see *The X.25 Interface for Attaching IBM SNA Nodes to Packet Switched Data Networks General Information Manual*.

XRF terminal switching is limited to SNA terminals. XRF does not support non-SNA devices that communicate with SNA hosts through other NPSI functions, including:

- The protocol conversion for non-SNA equipment (PCNE)
- Packet assembly/disassembly (PAD)
- General access to X.25 transport extension (GATE)
- Dedicated access to X.25 transport extension (DATE)

XRF Operational and Management Requirements

Although you do not need additional personnel at your installation to support XRF, additional tasks are required for system programmers and operators. The MVS, IMS, and network operators must understand the XRF process, be alert to the possibility of a takeover, and be ready to communicate with each other at takeover. You might need to organize your system consoles so that the operators responsible for managing the active system can easily communicate with the operators of the alternate system.

The system programmer must initialize MVS, IMS, VTAM, and NCP by using XRF-related parameters and macros. If your user applications conform to IMS standards, they run in the XRF complex and receive the availability improvements offered by XRF. System programmers can use standard diagnostic aids in an XRF complex.

Before installing XRF at your installation, you must prepare your data sets for minimum disruption during takeover. Make sure that:

- All required data is duplicated on active and alternate systems—that is, duplicate MVS data sets and non-volatile application and subsystem data.
- Access to application databases and logs is through DASD shared by the active and the alternate systems.

XRF does place additional demands on your installation's resources. However, if you understand how XRF works and plan carefully, you can reduce this overhead. The effort can be worthwhile if your installation is seriously affected by lengthy planned or unplanned interruptions of IMS service.

How Does Each Licensed Program Contribute to the XRF Process?

Five licensed programs perform the XRF process. As you plan, install, and operate XRF, and diagnose any problems that occur, you need to know the functions each performs.

Contribution of IMS

IMS is the major contributor to the XRF process, just as the IMS user is the greatest receiver of XRF service. While most of the IMS processing in the active system is serving the requests of the IMS terminal users, all of the IMS processing in the alternate system is related to XRF. The alternate IMS system refuses to accept logon requests from users. The following list summarizes the activities of the two IMS subsystems before and during a takeover.

- The active and alternate IMS establish and maintain constant communication through the IMS system log and through simple signals that the alternate IMS receives from active IMS. Surveillance mechanisms provide the signals.
- The active IMS:
  - Processes the requests of the IMS users
  - Sends surveillance signals across the ISC link and the RDS
  - Opens and closes sessions for terminals
  - Opens and closes databases
  - Reports its activities in the log
- The alternate IMS:
  - Responds to normal startup procedures, but does not process transactions until after a takeover
  - During normal operations:
    - Uses information from the log to update its control blocks
    - Allocates and opens the same databases that the active IMS opens
    - Opens and closes backup sessions for Class 1 terminals that log on to and log off from XRF IMS
    - Through surveillance mechanisms, monitors the active IMS and initiates takeover if surveillance indicates that useful work in the active system has ceased
  - During the takeover:
    - Participates with the other licensed programs in taking over the active system's workload. It obtains the workload by taking control of the IMS system log.
    - Prevents the failing active IMS from writing to the log.
    - Recovers the in-flight transactions, those transactions that the active IMS only partially processed.
    - Starts executing new transactions.
    - Starts session recovery [11] on Class 1 and Class 2 terminals.

IMS as a Recoverable Service Element

The two IMS subsystems in the XRF complex have a special status in the complex. Although failures in IMS, MVS, VTAM, and the CPC can cause takeovers and are therefore recoverable, XRF considers only IMS the recoverable service element (RSE). MVS, VTAM, and the CPC are dependent service elements (DSEs); they depend on IMS to recognize their failures and then to request a takeover. The RSE name appears in messages that the operators receive at takeover. For example, in Figure 17 on page 172, two IMS subsystems, named IMS1 and IMS2, form the RSE named IMSPROD. After a takeover at this complex, an MVS message tells the operators that `SUBSYSTEM IMS2 IS NOW THE ACTIVE ELEMENT OF RSE IMSPROD` (message AVM007I).

For more information about the RSE name, see RSENAME in the section "Using IMS.PROCLIB Members" on page 218.

11. "Session recovery" in this book refers to IMS attempts to switch sessions on Class 1 terminals or to reestablish service on Class 2 terminals. In the case of BTAM terminals, session recovery means reconnecting the terminals that were in use during takeover.

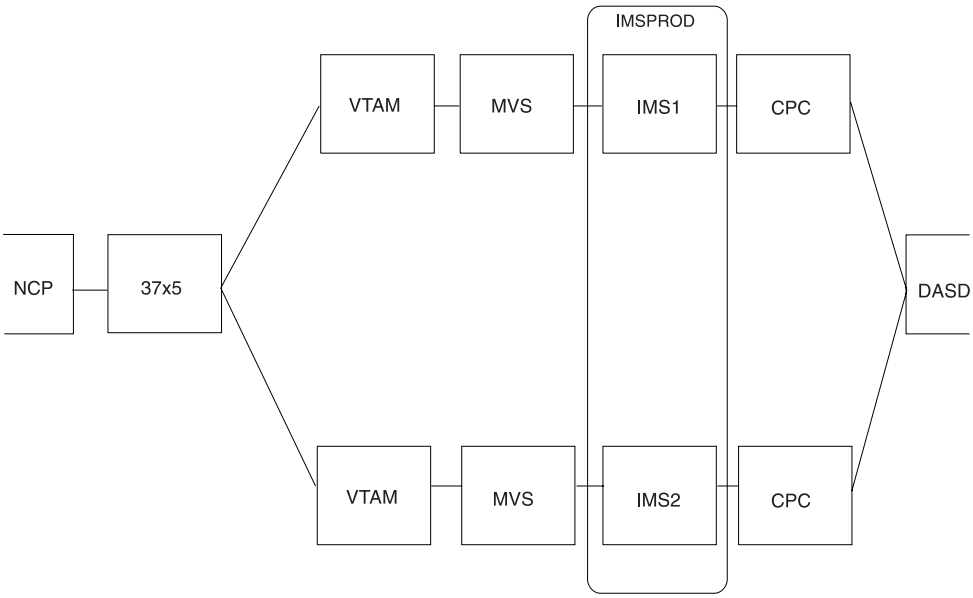


Figure 17. Naming the RSE

Surveillance Mechanisms

During normal operations, the alternate system tracks the active system by reading the IMS system log. The active system writes its activities on the log, and the alternate system checks these records, updating its own control blocks. In this way, control blocks in the alternate system reflect those in the active system, and the alternate IMS remains ready to take over.

The active IMS records on the system log the following failures:

- IMS control region abend
- VTAM failures that lead to TPEND exits in IMS
- IRLM failures that lead to STATUS exits in IMS

Certain error conditions can prevent IMS from recording these failures on the system log. An example of such a condition is the inability of IMS to purge the OLDS buffer.

The log does not warn of other failures. For this reason, surveillance mechanisms provide periodic signals from the active IMS. The alternate IMS monitors these signals. When the signals fail to appear, the alternate IMS considers a takeover. Absence of these signals can indicate:

- MVS failures
- MVS loops or wait states
- CPC failures

The alternate IMS can receive these signals in three ways:

- The active IMS sends messages over the ISC link between the active and the alternate IMS subsystems.
- The active IMS places a time stamp in the RDS.
- The active IMS continues to add new records to the IMS system log.

Using the parameters in member DFSHSBxx of IMS.PROCLIB, you choose which methods of surveillance operate in your installation. Figure 18 on page 173 shows

the three surveillance mechanisms. The ISC link can be a VTAM CTC or a shared line through the 37x5 Communication Controller.

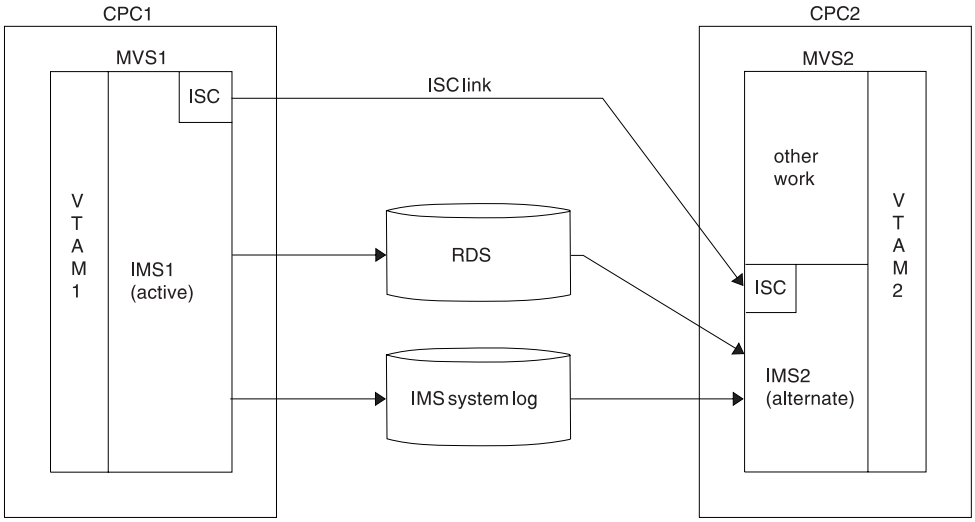


Figure 18. Surveillance Options

The signals across the ISC link and on the RDS are periodic. To use the IMS system log as surveillance, the alternate IMS periodically checks for new records on the log.

A failure to receive the signals on the ISC link and the RDS data set can cause a takeover; an absence of new records on the log cannot. However, if the alternate IMS knows of log activity, it overrides a takeover indication from the RDS or the ISC link.

In parameters you define in the DFSHSBxx member of IMS.PROCLIB, you specify:

- Which surveillance mechanisms you want your installation to use
- How often the active IMS is to send signals on the ISC link or the RDS
- How often the alternate IMS is to check the log for a new record
- Which absences of signals within specified times are takeover conditions

The next section, “Establishing Surveillance”, describes how to code these parameters. Your operator can stop and start surveillance dynamically by using IMS `/STOP` and `/START` commands. The operator can use the `/CHANGE` command to change the surveillance timing intervals.

Establishing Surveillance

To establish surveillance, perform the following four steps:

1. Determine which surveillance mechanisms to use for the complex.
2. For each surveillance mechanism you choose, decide the interval value (how often the alternate system is to receive the signals).
3. Decide whether the absence of any or all of these signals should be a takeover condition.
4. If the answer to Step 3 is yes, decide the timeout value (the length of time the alternate system waits for a surveillance signal before considering a takeover).

Five parameters determine surveillance: SURV, LNK, LOG, RDS, and SWITCH. The table below describes the planning decisions, the corresponding parameters, and the options on the parameters. The surveillance parameter and options on the parameters refer to:

- LNK as the ISC link between the two systems
- LOG as the IMS system log
- RDS as the RDS

Table 29. Surveillance Options on Parameters

Planning Decision	Parameters	Options
What surveillance mechanisms should operate in the XRF complex? (Step 1 on page 173)	SURV	LNK,LOG,RDS
If you specified LNK as surveillance, how long is the interval between signals? (Step 2 on page 173)	LNK	interval
If LNK is a takeover condition, how long should the alternate IMS wait for a signal before considering taking over? (Step 4 on page 173)		,timeout
If you specified LOG as surveillance, how long is the interval between signals? (Step 2 on page 173)	LOG	interval
If LOG is a takeover condition, how long should the alternate IMS wait for a signal before considering taking over? (Step 4 on page 173)		,timeout
If you specified RDS as surveillance, how long is the interval between signals? (Step 2 on page 173)	RDS	interval
If RDS is a takeover condition, how long should the alternate IMS wait for a signal before considering taking over? (Step 4 on page 173)		,timeout
Should the absence of specific surveillance signals cause the alternate IMS to request a takeover? (Step 3 on page 173)	SWITCH	LNK,LOG,RDS

Choosing the Surveillance Mechanisms

Use the SURV parameter to establish the surveillance mechanisms. You can specify combinations of the three mechanisms: LNK, LOG, and RDS. Because of the importance of surveillance, however, you should use all three mechanisms.

The alternate IMS uses these three surveillance mechanisms in different ways. Failure of the alternate IMS to receive signals on the ISC link or the RDS is possible cause for takeover; failure of the alternate IMS to receive new records on the log is not. Rather, IMS uses the LOG option to confirm or override a decision that the alternate IMS might make based on lack of signals from the RDS or the ISC link. For example, suppose you specify SURV=(LNK,LOG), and the alternate IMS stops receiving signals on the ISC link. The failure of signals over the ISC link indicates a takeover, but as long as the alternate IMS continues to receive log records satisfactorily, it does not request a takeover. In this case, IMS assumes that the ISC link itself is experiencing difficulty and is therefore not a reliable indicator.

If you do not have an ISC link already in place, establish it through the 37x5 Communications Controller that controls your Class 1 terminals.

For more information on surveillance mechanisms, see the section “Coding Parameters in DFSHSBxx” on page 219.

Setting the Interval Value

After you have established which surveillance mechanisms are to operate in the complex, you should decide how often the alternate IMS should receive signals through the surveillance.

Code the LNK and RDS parameters to establish the intervals between signals for the LNK and the RDS for each of the surveillance mechanisms you choose.

Two factors to consider in setting the timing for LNK and RDS are:

- The speed of communication over the ISC link.
- The performance overhead on the alternate MVS. Set the timing interval higher if you need to reduce the work of the alternate MVS.

Code the LOG parameter to establish how often the alternate system should check the log for new records.

For more information about interval values, see the section “Coding Parameters in DFSHSBxx” on page 219.

Related Reading: IMS has specific rules and recommendations for coding the interval values for the two systems. See *IMS/ESA Installation Volume 1: Installation and Verification* for these rules.

Specifying Surveillance Signal Absence as Takeover Condition

Use the SWITCH parameter to establish absence of surveillance signals as criteria for takeover. Specify on the SWITCH parameter the surveillance mechanisms that are critical in alerting the alternate IMS of certain events. For example, an installation managing its local locks on the active IMS with the IRLM, or one using data sharing, would specify the IRLM failure as a criterion for takeover. Installations with a large VTAM network would probably specify a takeover to occur when a VTAM failure causes an IMS TPEND exit. An installation with a large BTAM network and a small VTAM network would probably not specify a VTAM failure as a criterion for takeover. For more information about the SWITCH parameter, see the section “Coding Parameters in DFSHSBxx” on page 219.

Setting the Timeout Value

After you have specified which surveillance signal absences are takeover conditions, you can decide how long the alternate system should wait for a signal before requesting a takeover. Specify timeout values on the LNK, LOG, and RDS parameters. The default intervals are 9 seconds for the LNK parameter and 3 seconds for LOG and RDS.

If timeout values are too low, some MVS recovery routines that are transparent in a non-XRF environment might cause takeovers. An example of such a routine is an alternate CPU recovery (ACR) routine. These unwanted takeovers are more likely to occur on a larger CPC or with a large IMS workload.

Changing the Surveillance Mechanisms

The surveillance mechanisms and the internal parameters can be changed dynamically by commands issued from the master terminal.

Contribution of MVS

Through its availability manager component, MVS provides the environment for the active and the alternate system and provides services during a takeover. Specifically, the availability manager:

- Prevents the failing IMS subsystem from accessing the databases

  This action, called I/O prevention, introduces new procedures for your operators. This section describes I/O prevention and provides some of the messages that your operators might see.
- Sends messages to operators during the takeover

  These messages have the prefix, AVM. They describe the status of the XRF complex and help the operators respond correctly to the takeover.

The system resource manager (SRM) component of MVS hastens the acceptance by the alternate system of the new workload when a takeover begins. At this time, the alternate system temporarily requires more real storage to recover databases and sessions. The SRM component of MVS speeds up its analysis of the alternate system's need for storage. This frequent checking allows SRM to respond quickly to this need.

I/O prevention is the action the availability manager takes to stop the failing IMS from writing to the databases. It begins when the active MVS learns that IMS is requesting a takeover. At this time, the active IMS might have scheduled updates to the databases. To maintain database integrity, the availability manager in the active MVS ensures that current I/O operations are complete and that MVS does not honor any additional I/O requests to the databases from the active IMS. When the availability manager is sure the database data sets are safe, it issues a message that announces `I/O PREVENTION IS COMPLETE` (message AVM006E).

Of course, MVS cannot prevent I/O if it is no longer operating. If MVS cannot prevent I/O, an operator of the failing active MVS must manually make sure that the active IMS does not write to the databases.

Related Reading: For a description of operator's procedures, see *IMS/ESA Operations Guide*.

When an operator at the failing active system is certain that IMS is no longer changing the databases, that operator informs an operator at the alternate system. Then the operator at the alternate system responds "GO" to the message that says `REPLY` "GO" WHEN I/O `PREVENTION COMPLETES` (message AVM006E). This action (or the `/UNLOCK SYSTEM` command) completes the alternate system's efforts to ensure the integrity of the databases.

Your system programmers and operators must understand the importance of preventing IMS in the failing active system from accessing the databases. The integrity of the databases is lost if both IMS subsystems can write to them at the same time.

Related Reading: For information on establishing takeover procedures for your operators, see *IMS/ESA Operations Guide*.

Contribution of DFSMS

XRF needs DFSMS for I/O prevention and for VSAM and media manager. DFSMS, however, does not produce any messages or change any existing procedures.

Contribution of the Network Licensed Programs

The network licensed programs perform new functions for the users of terminals. Specifically, VTAM allows:

- The alternate IMS to initialize a backup session for a Class 1 terminal that has a session with the active IMS.
- NCP to switch backup sessions to primary sessions at takeover.
- A user to log on to XRF IMS using a single logon message. The user has no need to know which one of the IMS subsystems in the XRF complex is currently active.

NCP maintains the control blocks for the backup sessions and performs the switching of sessions when the alternate IMS requests it.

The SSP defines and generates an NCP Version 4. It also allows the user to load the program into the communication controller and dump the contents of the 37x5 back to the host CPC for diagnostic purposes. SSP does not produce any messages or change any existing procedures.

Your installation probably uses the services of the licensed program Network Communications Control Facility (NCCF). Although not modified for XRF, NCCF can signal a change in the VTAM application name to VTAMs inside and outside the XRF complex. If a VTAM in the network is not at Version 4 with the USERVAR management enhancement, an NCCF CLIST should be used to propagate a USERVAR change to that VTAM. This change in the VTAM application name occurs at initialization—at completion of IMS restart or processing of the `/START DC` command—or at a takeover. If your installation does not have NCCF, or if a VTAM is below Version 4, operators in each system in the network must issue the `MODIFY` command to update the VTAM application name.

Backup Sessions for Class 1 Terminals

To reduce the time necessary to switch the terminals at a takeover, the alternate IMS opens backup sessions for Class 1 terminals. Whenever a Class 1 terminal user logs on to XRF IMS, the active IMS records this new session on the log. The alternate IMS reads this record and initializes a backup session for the terminal. Data does not flow on the backup session as long as it remains a backup session. When the user logs off from XRF IMS, the alternate IMS terminates the backup session.

At a takeover, the alternate IMS requests, through VTAM, that NCP switch each of the Class 1 terminals that are communicating with the active IMS from primary sessions to the preopened backup sessions. NCP responds to these requests. Transparency of takeover depends on what is happening on the terminal at the moment of takeover. The user might be unaware of the takeover or might be asked to reenter the last transaction.

Terminal Logon in the XRF Complex

With XRF, the two IMS subsystems in the complex appear to terminal users as a single IMS subsystem. However, to VTAM they are unique applications. VTAM allows a terminal user to log on to XRF IMS with a logon message that you choose. (Or, a terminal user can log on using the command `LOGON APPLID` that specifies the logon message.) VTAM checks its interpret table for the variable that corresponds to the logon message. VTAM then looks in the USERVAR table for the VTAM application name that corresponds to this variable. This application name identifies the IMS subsystem for the terminal to communicate with at this time. This IMS subsystem is the session partner for the terminal.

Be sure that you understand the two tables and the entries in the tables. The variable that VTAM uses to associate the logon message with the application name of the session partner is called USERVAR. It resides in both the interpret table and

the USERVAR table. Figure 19 illustrates the interpret table and the USERVAR table. The two entries in the interpret table allow users to log on to XRF IMS with the logon messages IMSP or IMSA.
The interpret table is a static table that you build at VTAM initialization. You use the

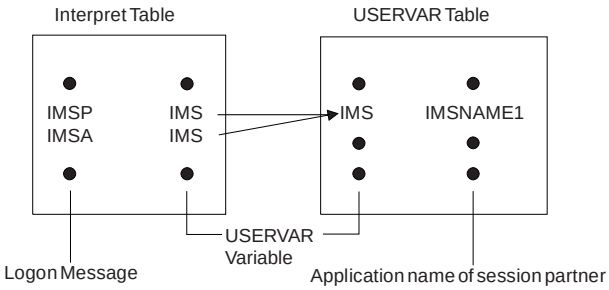


Figure 19. The Interpret and USERVAR Tables

LOGCHAR macro to place entries in this table. Entries in the USERVAR table are established at initialization but can be changed dynamically. IMS issues the `MODIFY USERVAR` command to initially place an entry in the USERVAR table. An operator or an application program uses the same command to delete a user-managed USERVAR and specify that VTAM manage the USERVAR automatically, or to change an entry in the USERVAR table. For example, at takeover, the `MODIFY USERVAR` command changes the application name in the USERVAR table to reflect the new session partner for all terminals.

Related Reading: For more information on the VTAM USERVAR table, see "VTAM USERVAR Table Definition" on page 210.

The top part of Figure 20 on page 179 shows the interpret and USERVAR tables for the VTAM that owns the terminals. In this case, it is VTAM in the alternate system. When a user logs on with the logon message IMSP, VTAM searches the interpret table for IMSP and finds the corresponding USERVAR IMS. VTAM then searches the USERVAR table for the application name that corresponds to IMS and opens the session with the IMS subsystem IMS1.

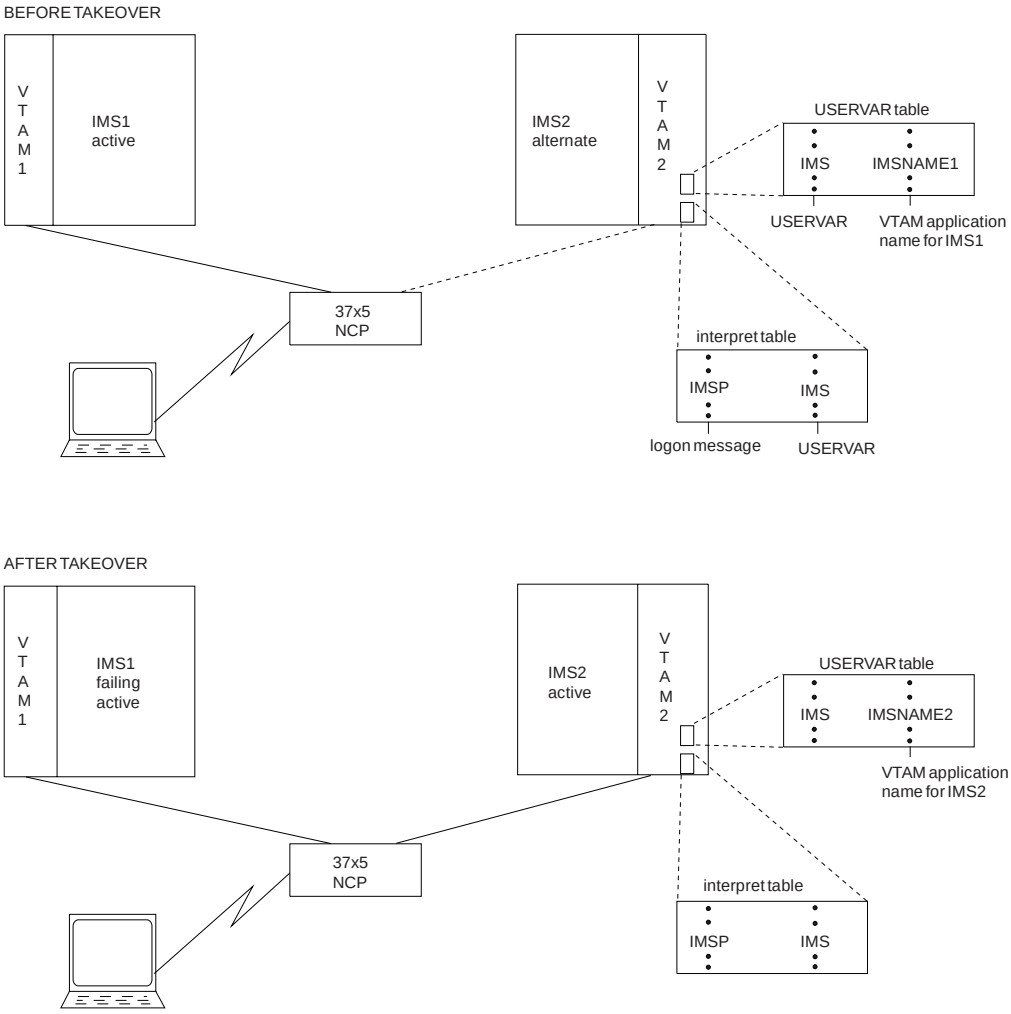


Figure 20. VTAM Processing of the Logon

The bottom part of Figure 20 shows that, following a takeover, VTAM2, at the request of IMS2, has changed the application name in the USERVAR table. Now when a user at a terminal specifies IMSP, VTAM connects the user to IMS2.

All existing user programs that conform to present IMS standards continue to execute in the XRF complex. If the USERVAR is VTAM-managed, a modification to the new VTAM APPLID is done automatically. If you have programs still using the logon that they used previously and that are not VTAM-managed, you must make a modification, because the logon does not match the VTAM APPLID of the active IMS.

To modify application programs with user-managed USERVARs, do the following:

- Use INQUIRE USERVAR to communicate with the active IMS to determine the real name of the current active system.
- For terminal programs sensitive to the PLU/SLU name in BIND data, the USERVAR is appended by IMS in BIND data.
- New VTAM SENSE codes need to be tested.

IMS prevents a user from logging on to the alternate IMS with the message `INVALID LOGON REQUEST IN THE BACKUP SYSTEM` (message 3862I). Only IMS master

and secondary terminals, the system console, and the ISC surveillance link can communicate with the alternate IMS. The term BACKUP in this message refers to the alternate IMS in the XRF complex. In IMS messages, the term BACKUP can also refer to the backup sessions on Class 1 terminals.

Phases of the XRF Process

The two systems that form your XRF complex pass through six identifiable phases. The phases of XRF are:

- Initialization
- Synchronization
- Tracking
- Takeover
- Post-takeover
- Termination

You can determine the current phase at any time by checking the system status line on the console for the alternate IMS.

Each subsystem participates in each phase, but the active IMS has little XRF activity in the synchronization, tracking, and takeover phases. The following sections describe the activities for the active and alternate subsystems for each of the six phases.

Initialization

The initialization phase begins when an operator brings up the availability manager as an MVS started task, either through the START AVM command or through the /NRESTART or /ERESTART command. The START command loads modules and control blocks and, optionally, opens a session on the ISC link that connects the active and the alternate systems.

See Figure 21 on page 181 for the .

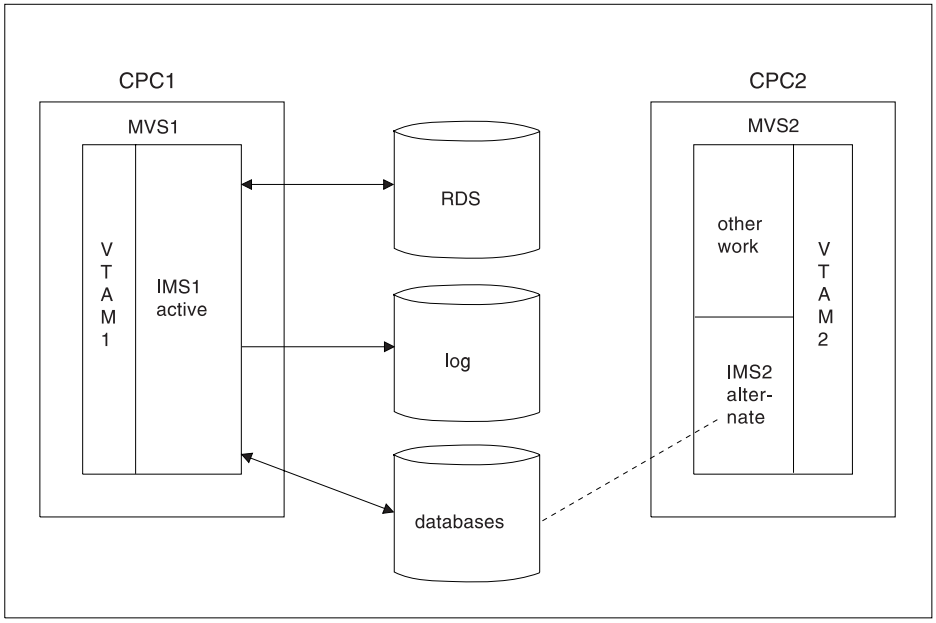


Figure 21. XRF Initialization Phase

An operator at the active system uses MVS START commands to bring up the active IMS, as in a normal or emergency restart. When it is up, the operator at the alternate system can bring up the alternate IMS at any time. The commands are similar for both systems except that the operator at the alternate system adds the option AUTO=N to the START IMS command and issues the IMS command, /ERESTART BACKUP.

Attention: An operator error can result in losing system or database integrity.

The alternate IMS is in a state similar to emergency restart until a takeover occurs. To be ready for a takeover, the alternate IMS shares the databases with the active IMS. The alternate IMS opens the databases, but does not access them until after a takeover.

In addition to the normal IMS initialization functions, IMS does the following to set up the XRF complex:

1. Processes the new DFSHSBxx IMS PROCLIB member. For more information, see “Coding Parameters in DFSHSBxx” on page 219.
2. Loads the new XRF/IMS modules, and obtains the new pools and work areas. Most of the new modules reside in the control region private below the 16 MB line.
3. Identifies both subsystems to the MVS availability manager using the RSENAME keyword, described in section “Using IMS.PROCLIB Members” on page 218.
4. Establishes the DBRC environment on both the active and the alternate subsystems using the RSE name. If a takeover occurs, authorizations of the active IMS are automatically passed to the alternate IMS.
5. Informs VTAM of the APPLID of the active IMS subsystem.

Related Reading: For more information on how the operators bring up the active and alternate IMS subsystems, see *IMS/ESA Sample Operating Procedures*.

Synchronization

During the synchronization phase, the alternate IMS becomes an image of the active IMS, as shown in Figure 22. The phase begins when the environments are fully initialized.

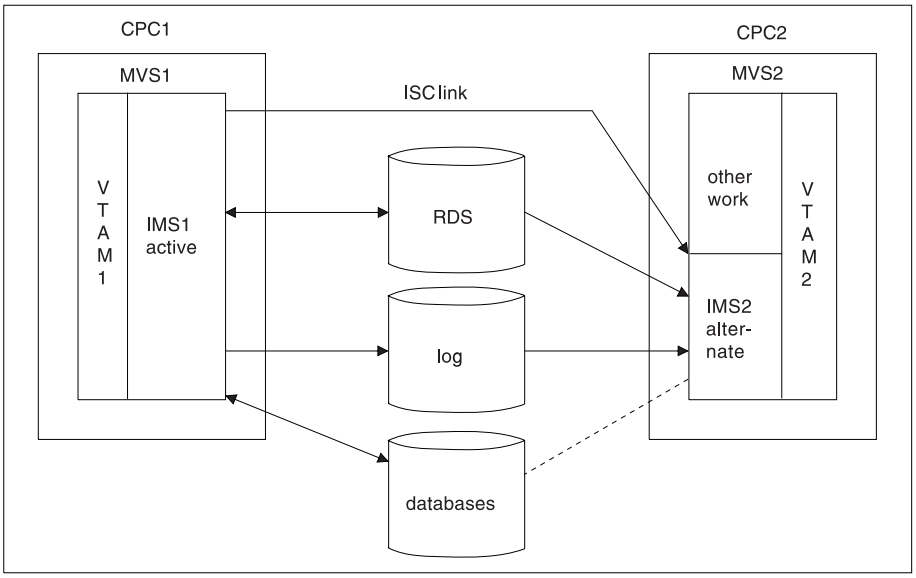


Figure 22. XRF Synchronization Phase

The process of building the control blocks in the alternate IMS is very simple. The alternate IMS determines if the last checkpoint was a SNAPQ. If it was not, and you have an ISC link, the alternate IMS requests that the active IMS take a SNAPQ checkpoint. The SNAPQ log records are used to check that the IMS system definitions on both systems match and to build an initial “snapshot” of the active IMS on the alternate IMS.

If you do not have an ISC link, the alternate IMS issues a message to ask the master terminal operator at the active IMS to request the SNAPQ checkpoint.

The alternate IMS opens the IMS system log, locates the SNAPQ checkpoint, and reads from the log data sets.

If the ISC link exists, a message is sent to the alternate IMS while the checkpoint is being written to the log. If the ISC link does not exist, the alternate IMS monitors the RDS for the completion of SNAPQ.

IMS uses the contents of the dump for several things:

- The alternate IMS determines which MODSTAT and RDS are being used by the active IMS.
- The alternate IMS tries to preallocate and pre-open databases and areas that are accessed by the active IMS.
- MSDBs are loaded by the alternate IMS from the SNAPQ checkpoint records; separate MSDB checkpoint data sets must be available for the alternate IMS.
- Backup sessions for terminals opened on the active IMS are initiated on the alternate IMS for those Class 1 terminals that are defined with the BACKUP option.

- Security control blocks are loaded on the alternate IMS from the IMS.MATRIX data set.
- The DBRC subsystem record is updated to indicate that an alternate subsystem exists. This suppresses the DBRC `'SIGNOFF ABNORMAL'` message in the case of failure of the active IMS.
- Status of dependent regions on the active IMS is tracked on the alternate IMS.

Only after a takeover can the alternate IMS write to the log.

Operators can start dependent regions in the alternate IMS to have them ready for a takeover.

When the alternate IMS has processed the SNAPQ checkpoint, the synchronization phase is complete. When the alternate IMS's control blocks are synchronized with the active IMS's, the alternate IMS is able to take over the processing of the active IMS. From this point on, you can request a planned takeover.

Tracking

Most of the time, your XRF complex is in the tracking phase, shown in Figure 23. The active IMS processes the requests of IMS users, writes to the log, and sends signals across the surveillance mechanisms that were set up in the initialization phase. The alternate IMS tracks the active IMS to reduce the work that the alternate IMS must do at takeover.

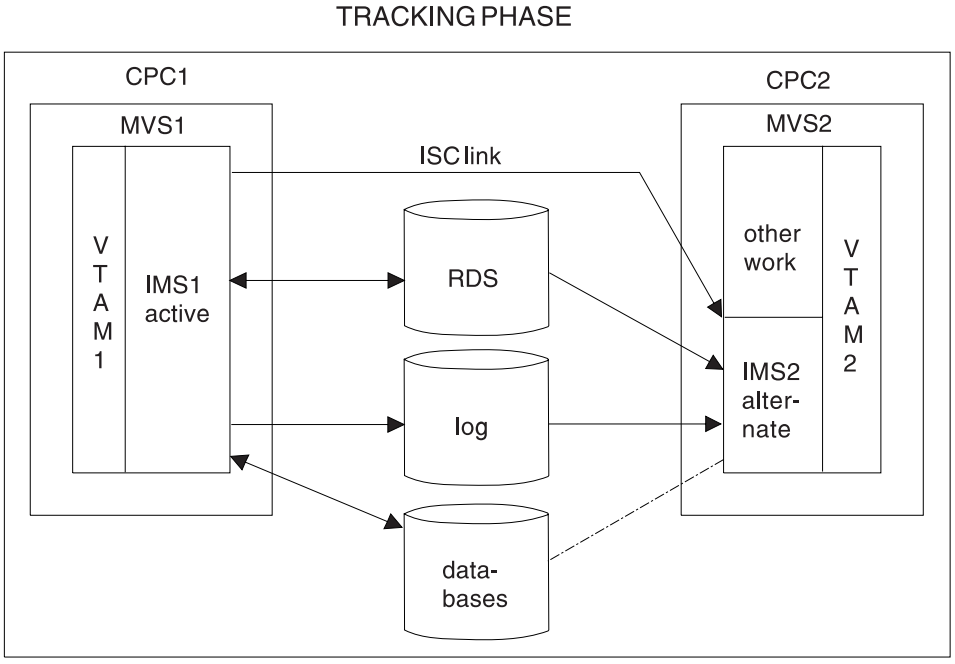


Figure 23. XRF Tracking Phase

Tracking activities of the alternate IMS include:

- Reading the log
- Using log information to update control blocks
- Opening and closing backup sessions for Class 1 terminals
- Checking the surveillance mechanisms and the log records for signs of failure in the active IMS

Tracking activities use few of the non-storage resources of the alternate IMS. The use of real storage depends on how you page fix IMS storage and whether you define a member of DFSFIXxx with page-fixing options that are in effect only when the subsystem is active.

Updating Control Blocks in the Alternate IMS

The alternate IMS reads the OLDS [12] to track:

- Communication Network Status

  In addition to the usual terminal status tracking that is done during the emergency restart process, the alternate IMS monitors the terminal connection status. This is done initially from records in the SNAPQ checkpoint and subsequently from terminal connection status change log records written by the active IMS. For Class 1 terminals, backup sessions are maintained by the alternate IMS. When terminal sessions are terminated on the active IMS, they are also terminated on the alternate IMS. The user can never log onto the alternate IMS. Terminal types are described in “Terminals in an XRF Complex” on page 202.

- Application Program and Transaction Status

  As application programs are scheduled on the active IMS, the alternate IMS builds the control block structure to track the dependent region status.

  The operator can prestart dependent regions on the alternate IMS to allow the alternate IMS to complete takeover more quickly.

- Database Status

  As databases and areas are allocated and opened on the active IMS, the alternate IMS pre-allocates and pre-opens these same databases and data set areas. The IMS macro definitions for all databases and areas must be specified as SHARED. The alternate IMS also tracks all DBRC database authorizations. It is recommended that you register all databases and areas in DBRC. The alternate IMS also tracks the status of all locks for uncommitted database changes.

- Message Queue Status

  Message queues are built on the alternate IMS from the SNAPQ checkpoint records. They are updated on the alternate IMS by OLDS records. A local message queue is used by the alternate IMS to communicate with the operator. At IMS system generation time, two master and two secondary master terminals are specified. The master terminals at the active IMS control the active IMS; the master terminals at the alternate IMS control the alternate IMS.

- MFS Pool Status

  MFS blocks are preloaded and released on the alternate IMS as they are loaded and released on the active IMS.

- Dependent Region Status

  The alternate IMS initializes and maintains control blocks to reflect the status of the dependent regions. It monitors all activity in the region so that backout of uncommitted processing can occur at takeover.

With this information, the alternate IMS updates its control blocks to maintain an environment that duplicates the active IMS. It pre-allocates and pre-opens the IMS databases and data areas as the log indicates that the active IMS allocates and opens them. It also loads the MFS control blocks, program specification blocks (PSBs) for backout use, and data management blocks (DMBs).

12. Use dual OLDSs, because tracking can continue using the other OLDS if one OLDS is lost.

Surveillance of the Active IMS

As described in “Surveillance Mechanisms” on page 172, the alternate IMS depends on surveillance to indicate some of the takeover conditions in the active IMS. If surveillance includes the RDS, the active IMS periodically places time stamps on the RDS. If surveillance includes the ISC link, the active IMS sends periodic signals over the ISC link. If surveillance includes the log, the alternate IMS periodically checks to ensure that log records continue to arrive.

Takeover

The alternate IMS requests a takeover for one of three reasons:

- A problem occurs that is so serious that processing in the active IMS cannot continue. These problems include:
  - An IMS abend
  - An MVS failure, loop, or wait state
  - A CPC failure

  These problems are based on the assumption that you establish at least basic surveillance including the RDS and the ISC link. If surveillance detects a failure in the active IMS, or if the operator issues a `/SWITCH SYSTEM FORCE` command, the alternate IMS reserves the logs to prevent the active IMS from furt
- The operator initiates a planned takeover.

  To initiate a planned takeover, an IMS operator at the active or alternate IMS issues the `/SWITCH SYSTEM` command to cause a takeover. IMS waits until all in-flight messages are completed for every dependent region before it abends. The availability manager in the old active system performs I/O prevention, and no backouts have to be performed on the alternate IMS. An example of the use of the planned takeover is when your system programmer wants to do maintenance in the active system. “Practical Uses of the Planned Takeover” on page 191 suggests other uses of a planned takeover.
- In parameters in member DFSHSBxx of IMS.PROCLIB, you establish other criteria for takeover.

  As mentioned in “Surveillance Mechanisms” on page 172, you can specify that an absence of a signal from surveillance causes a takeover. For example, you can tell the alternate IMS that the absence of a signal over the ISC link for ten seconds is a takeover condition.

  Two other failures that you can set up as criteria for takeover are:

  – Abend of the IRLM

    You have IRLMs at your complex for two reasons:

    - The IRLM in the active system might control access to the databases that XRF IMS shares at a block level with IMS subsystems outside the complex. In this case, your XRF complex has two IRLMs, one at the active subsystem and one at the alternate subsystem.
    - IRLMs might be managing your local locks.

    In either case, you might want a takeover to occur if the IRLM in the active system terminates, at which point, an IMS STATUS exit routine is invoked.

  – Failure of VTAM

    You can direct the alternate IMS to request a takeover if VTAM invokes the IMS TPEND exit routine.

  Failures of VTAM and the IRLM cause degraded performance at your installation, but you might still decide to avoid a takeover if these failures occur.

You specify the absence of surveillance signals or VTAM or IRLM failures as takeover conditions in the parameters you add to member DFSHSBxx of IMS.PROCLIB. “Establishing Surveillance” on page 173 describes these parameters.

The alternate IMS checks the signals it receives on the RDS and the ISC link, reads the records it receives on the log, and monitors the commands of the operators. Eventually it decides to request a takeover. At this point, the takeover begins. Depending on how IMS is tailored, the takeover proceeds automatically or waits for an operator to perform certain tasks before allowing the takeover to proceed. The takeover phase is illustrated in Figure 24.

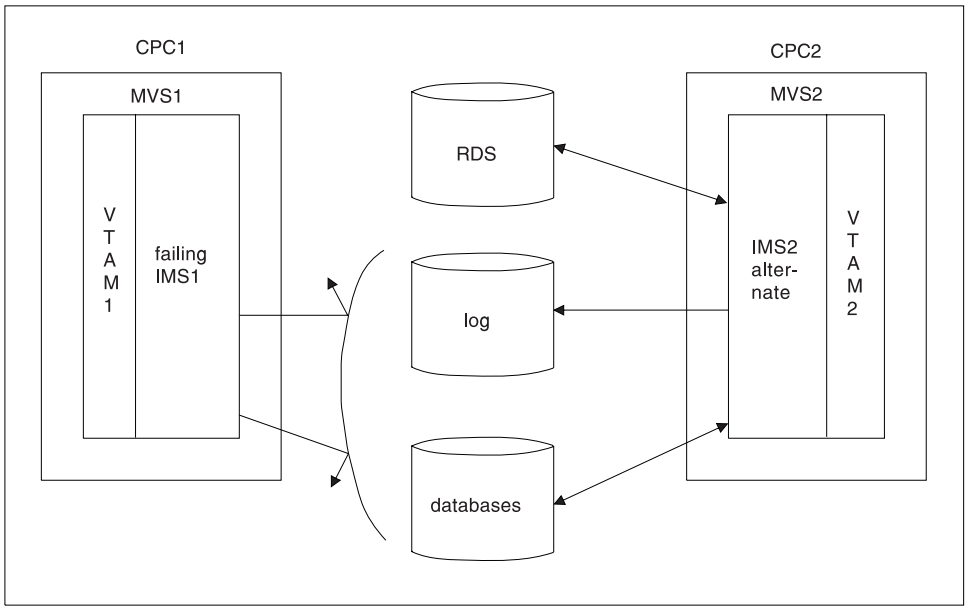


Figure 24. XRF Takeover Phase

During takeover, the alternate IMS uses the data it collected during the synchronization and tracking phases to transform itself into an active IMS. It also ensures that work on the former active subsystem halts. XRF’s primary objective during this phase is to ensure the integrity of the databases. At the end of this phase, the alternate IMS is the new active IMS. After the takeover begins, it cannot be revoked—it proceeds to completion.

The Takeover Begins

Because the condition of the failing active IMS at takeover is uncertain, the alternate IMS does most of the work at takeover. The alternate MVS must decide what to do with the non-XRF workload that was processing on the alternate system. See “Non-XRF Workload on the Alternate IMS” on page 201.

The following events occur during a takeover if IMS abends or if an operator at the active IMS issues the `/SWITCH SYSTEM` command. “Differences in the Takeover Process” on page 189 describes the takeover process caused by other events.

During a takeover, the active IMS attempts to do the following:

- Recognizes the failure and writes a termination message to the log. It stops processing incoming messages.
- Issues a RESERVE against the OLDS and WADS of the active IMS if the active and the alternate subsystems reside on different CPCs. It is recommended that

no other data sets reside on these devices, because they are inaccessible after the RESERVE is issued. Access to these data sets when a takeover has begun might also hold up the takeover.

If the active and the alternate IMS subsystems reside on the same CPC, the activities differ: the alternate IMS must ABTERM the active IMS and wait until the IMS resource clean-up routine on the active IMS notifies the alternate IMS that I/O to the log has terminated on the active IMS.

- Closes the OLDS using the WADS when IMS can no longer access the logs. Because a takeover is not viewed as a failure by DBRC, the alternate IMS switches to the next set of previously allocated OLDS.
- Invokes the MVS system resource manager (SRM), which tries to obtain resources for the alternate IMS. Because the alternate IMS temporarily requires more real storage to recover databases and sessions, SRM hastens the acceptance of the new workload by the alternate IMS.
- Removes incomplete messages during takeover. These incomplete messages occur at takeover when the message queues have already been built on the alternate IMS.
- Defers the deletion of nonrecoverable messages.
- Re-enqueues recoverable messages that were in process at the time of failure, unless they must be backed out. If backout is required, these messages are automatically re-enqueued at the completion of each backout.
- Merges messages on the local message queue so that the IMS master terminal operator does not lose critical messages due to the takeover.

The active IMS performs the listed tasks except when it cannot purge the OLDS buffer.

If the availability manager is operating in the active system, it performs I/O prevention. Normally, I/O prevention is completed within a few seconds of the I/O prevention request. “Contribution of MVS” on page 175 describes the action the availability manager takes to stop the failing IMS from writing to the databases. If the availability manager is not operating, the operator at the active system must manually perform I/O prevention by resetting the CPC or terminating IMS. At the completion of I/O prevention, an operator at the active system must inform an operator at the alternate system of the completion, and the operator at the alternate system must reply “GO” to message AVM005A.

Even before this notification, the alternate IMS proceeds with the takeover while it also protects the databases. It acquires the key shared resources by reserving the DASDs that contain the current OLDS and WADS.

After it reserves the log, the alternate IMS performs three activities in parallel:

- Recovers the data in the message queues and the databases
- Performs network changes
- Executes new and reprocessed transactions

The following sections describe these three parallel activities.

Recovering Data in Message Queues and Databases

To process the messages that the active IMS did not begin to process, the alternate IMS ensures that its own copy of the message queue is complete. It can then process these messages and update the databases.

To recover the in-flight transactions, the alternate IMS backs out of the databases all the transactions that the active IMS began but did not complete, and reprocesses the transactions. To ensure integrity of the Fast Path databases, it performs forward recoveries.

The alternate IMS performs I/O toleration by delaying writing to the databases the changes to the data block or VSAM control interval that could be overwritten by the failing active IMS until the alternate IMS is certain that the failing active IMS can no longer write to the databases. The alternate IMS keeps the changes temporarily in storage and makes all references to these changes to this storage area rather than to the database. The operators see the term `I/O TOLERATION` in status displays under the heading `MODE`. If the availability manager is not active on the failing active system, the operator at the active console must prevent I/O operations by canceling IMS or resetting the system.

The alternate IMS subsystem performs the following actions as part of I/O toleration:

- It identifies and tags database blocks or control intervals that were being updated at the time of failure with an Extended Error Queue Element (EEQE).
- It keeps these records in virtual buffers to enable access during dynamic backout and transaction processing until the user terminates I/O toleration. It also propagates the EEQEs to DBRC and notifies sharing subsystems.
- It backs out possibly incomplete updates that are in-flight and reschedules them for processing.
- It stops using any databases that are being extended at takeover.

IMS restricts access to any data that it shares (through the service of the IRLM) with an IMS subsystem outside the XRF complex. Specifically, it prevents other IMS subsystems from accessing the blocks of data that are affected by in-flight transactions.

When an operator at the alternate system replies “GO” to message AVM005A: `REPLY "GO" WHEN I/O PREVENTION COMPLETES` (or issues the `/UNLOCK SYSTEM` command):

- IMS updates the databases with the changes for in-flight data it kept in storage.
- IMS makes all portions of the databases available to the IMS subsystems outside the XRF complex.
- I/O toleration ends.

The IMS command `/UNLOCK SYSTEM` performs the same action as the response “GO”.

Initiating Network Changes

To allow additional terminal users to log on to XRF IMS after a takeover, VTAM must change the application program name in its USERVAR table. The alternate IMS issues a `MODIFY USERVAR` command to change the entry in its own USERVAR table. Network operators at all other VTAMs that communicate with XRF IMS must issue the `MODIFY` command to change the application program name in their USERVAR tables if this procedure has not been automated. This procedure can be automated through NCCF. When the alternate IMS issues the `MODIFY` command, NCCF signals the change in application program name to other NCCFs with which it is in session.

At takeover, three obvious changes occur on the terminals: sessions on Class 1 terminals switch to backup sessions, sessions on Class 2 terminals are reestablished, and sessions on Class 3 terminals stop. For Class 3, service is almost the same as if XRF were not present. “Terminals in an XRF Complex” on page 202 describes the different characteristics of Class 1, 2, and 3 terminals.

The alternate IMS requests that NCP switch each primary session on a Class 1 terminal to the preopened backup session. NCP cannot switch a session that does not have a backup session already open. If the failure in the active IMS has not already caused the session with the failing IMS to terminate, NCP then terminates the primary session.

How quickly a session switches to backup or is reestablished depends on the type of CPC in the alternate system, the switching priority of the terminal, the size of the network, and the number of NCPs performing the switches.

IMS tries to reestablish sessions with the Class 2 terminals, such as terminals that communicate with XRF IMS through MSC physical links.

Executing New and Reprocessed Transactions

The alternate IMS isolates the log from the active IMS and deletes uncommitted messages in the message queues. It then reacquires the locks needed to perform recovery and starts processing new and reprocessed transactions. As IMS recovers sessions on terminals and makes shared databases available, processing of new transactions increases.

At a takeover, unshared databases are available immediately. Shared full-function databases become available after completion of the Database Recovery Control (DBRC) reverify. Shared data entry databases (DEDBs) become available after completion of forward recovery and DBRC reverify, and after a simple checkpoint is taken.

Differences in the Takeover Process

The preceding pages describe the takeover process caused by the IMS abend and the `/SWITCH SYSTEM ACTIVE` command. For all other causes of a takeover, the availability manager does not perform I/O prevention. The IMS operator at the active system must perform one of two actions before notifying the IMS operator at the alternate system that the active system can no longer write to the database.

- If the CPC or MVS fails, the operator must reset the CPC.
- For all other causes of failure, the operator must terminate IMS to invoke I/O prevention.

Because there is no communication between the active MVS and the alternate MVS, an operator at the active MVS must inform an operator at the alternate MVS that the databases are safe.

IRLM Processing at Takeover

If your XRF complex shares its databases at the block level with one or more IMS subsystems, Figure 25 on page 190 shows an XRF complex that shares its databases with another XRF complex. In Figure 25 on page 190, the primary and secondary relationships refer to the IMS partners. In IRLM 2.1, there is no primary or secondary IRLM. Communication exists between all IRLMs in the sharing group. One complex consists of:

- An active IMS, IMS1, operating with IRLM1
- An alternate IMS, IMS2, operating with IRLM2

The second complex consists of:

- An active IMS, IMS3, operating with IRLM3
- An alternate IMS, IMS4, operating with IRLM4

When the operators start these complexes, the IMSs initialize these sharing

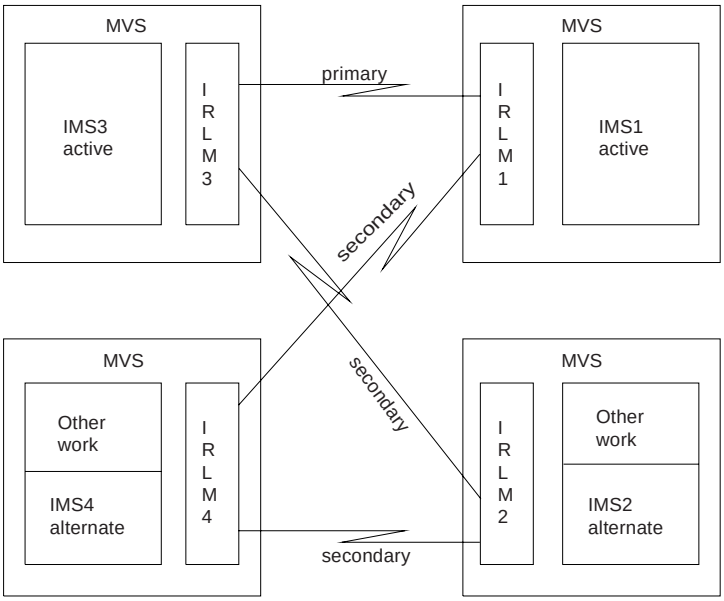


Figure 25. Two Data-Sharing XRF Complexes

relationships:

- Primary sharing partners IMS1 and IMS3
- Secondary partners between:
  - IMS3 and IMS2
  - IMS1 and IMS4

At a takeover caused by a failure of IMS1, IMS2 notifies IRLM2 of the takeover and tells the operator: `IRLM TAKEOVER ISSUED`. IRLM2 tells IRLM3 to prepare for takeover and IMS2 acquires the locks that were held by IMS1. When IMS2 completes its takeover processing, it assumes the identity of IMS1. IMS3 and IMS2 (alias IMS1) now become the primary sharing partners. IMS2 issues message DFS3887.

Once the takeover is complete, IMS1 can restart as the alternate IMS on IRLM1.

Data sharing at the block level adds to the post-takeover procedures for the IRLM operator. Before bringing up IMS1 as the alternate IMS, the operator must restart IRLM1 to establish two new secondary IRLM sessions between:

- IRLM3 and IRLM1
- IRLM4 and IRLM1

Related Reading: Similar procedures follow a takeover in the other complex. See *IMS/ESA Operations Guide* for descriptions of those procedures.

XRF IMS might use the IRLM as a local lock manager. In this case, the IRLMs do not communicate with each other. At takeover, the alternate system acquires its own locks and issues takeover messages.

Practical Uses of the Planned Takeover

Through the planned takeover, XRF gives you flexibility to perform some tasks during the day that in the past you had to do in evening hours or on weekends. A planned takeover begins when the operator at the active IMS issues a `/SWITCH SYSTEM ACTIVE` command and the workload on the active IMS transfers to the alternate IMS. The active IMS attempts to quiesce before the alternate IMS performs the takeover. Dependent regions are allowed to reach synchronization points. After the operator brings down the former active IMS, the system programmer can perform these tasks on the former active IMS:

- Testing of backup procedures
- Code maintenance
- Preventive maintenance
- Hardware maintenance
- Library maintenance
- Hardware configuration changes

After you have performed the task that you planned, bring up that system as the alternate IMS in the XRF complex.

You must use the planned takeover carefully; some hardware and IMS initializations must occur simultaneously in both environments. For example, because the active IMS and the alternate IMS must have the same system definition, you cannot request a takeover to change the IMS system definition while the system is the alternate. Unless you can make the system definition changes using online change, a graceful shutdown of both subsystems, followed by an IMS cold start, is required.

In general, you can apply most program temporary fixes (PTFs) without disrupting your end users. Exceptions are when you must terminate the XRF complex to apply the service in both systems simultaneously. These cases are identified in the PTFs. For IMS service, if the maintenance does not require an IMS cold start in a non-XRF environment or change the IMS log record format, you can apply the PTF while one of the systems is running.

Use the planned takeover to shift the workload to another environment when demands for IMS services increase or decrease. For example, you can run your production work on an ES/9000 CPC during the daytime period of peak activity. You can shift the workload to a 3083 during the nighttime period of lower activity. These two takeovers each day might be a valuable use of XRF for your installation.

Post-Takeover

When the takeover is completed, the alternate IMS becomes the new active IMS, as shown in Figure 26 on page 192.

The post-takeover phase overlaps the completion of the takeover phase. As the new active IMS processes new user transactions, IMS continues to perform DL/I backouts, Fast Path forward recoveries, and network switching. The operator can bring up a new alternate IMS after the new active IMS is told that I/O prevention is complete and the old active IMS has terminated.

The following events occur in parallel on the new active subsystem:

- The standard Fast Path DEDB forward recovery is completed. IMS reacquires locks for VSAM CIs containing unwritten but committed changes. An EEQE is created for each of these changes.

- DL/I restart database backouts are completed. The alternate IMS inherits the DBRC authorization for the databases from the previous active IMS instead of being closed during the restart.
- The new active IMS page fixes the non-Fast Path options identified by the DEFERFIX=xx parameter in DFSHSBxx.
- The new active IMS notifies the MVS availability manager that it is the active IMS. The operator of the failed active IMS must inform the operator of the alternate IMS when I/O prevention is complete so that I/O toleration on the alternate IMS can be discontinued.
- Service is returned to some Class 3 terminals.

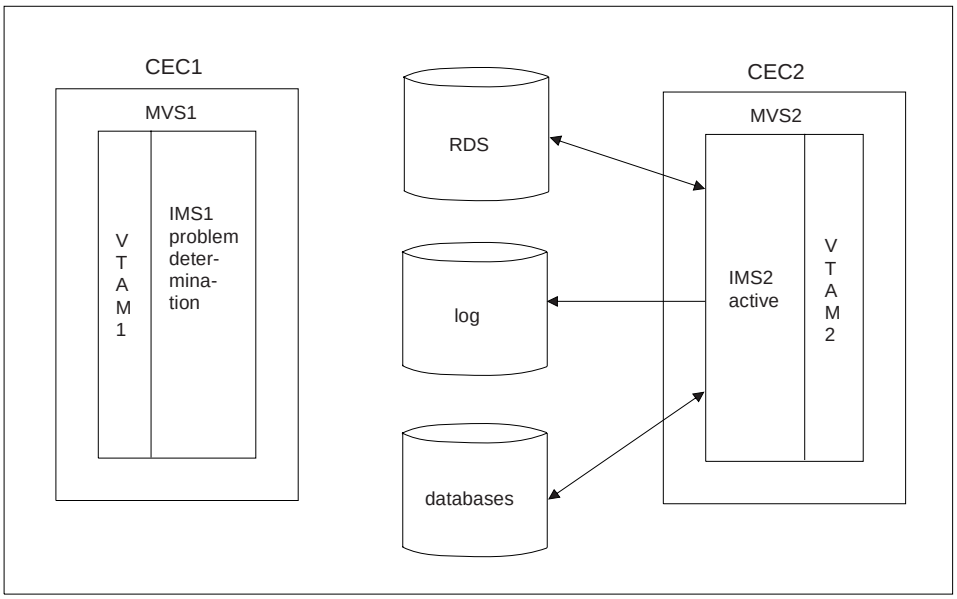


Figure 26. XRF Post-Takeover Phase

During the post-takeover phase, the new active IMS does not have an alternate environment. It has responsibility for providing service to the end users without benefit of XRF.

The failed IMS subsystem performs problem determination activities. When these are completed, the operator brings up the failed IMS as a new alternate IMS.

If problem determination activity requires lengthy dumping of the IMS data areas, you might consider bringing up a third system as the new alternate IMS in the XRF complex. See “Bringing up a Third System as an Alternate IMS” on page 193.

The post-takeover phase ends when an operator establishes a new alternate IMS—and thus a new XRF complex.

Returning the Failed Active IMS as the New Active IMS

Instead of leaving the complex in this arrangement, you might want to return to the original one. For example, if the failed CPC is superior to the other in speed of execution or storage capacity, you might want to restore it as the active IMS as quickly as possible. You cannot reinitialize the failed active IMS as a new active IMS; you must stage its return. First, bring it up as an alternate IMS and then manually request a takeover. This shift of the workload, initiated by the `/SWITCH SYSTEM` command, is another example of a planned takeover.

If XRF IMS shares its databases with other IMS subsystems, the takeover is a more complicated process that involves the efforts of the IMS operator. Procedures for the operator are described in “IRLM Processing at Takeover” on page 189.

Bringing up a Third System as an Alternate IMS

To limit the time the new active IMS is without an alternate IMS, your operator can bring up a third system as an alternate IMS. You can initialize the new alternate IMS as soon as I/O prevention is complete and the new active IMS issues message DFS994I to signal that it has taken its first checkpoint. At this time, the operator at the third system can issue the `/ERESTART BACKUP` command to start an alternate IMS.

You are responsible for ensuring that paths exist to all devices, including DASD and communication controllers.

Termination

The termination phase returns the XRF complex to two separate and independent environments and stops all activity in the alternate IMS. While a takeover is a termination in the sense that it removes the active IMS from service, termination as a separate XRF phase occurs when an operator stops one or both of the IMS subsystems.
The three types of termination of the active IMS are:

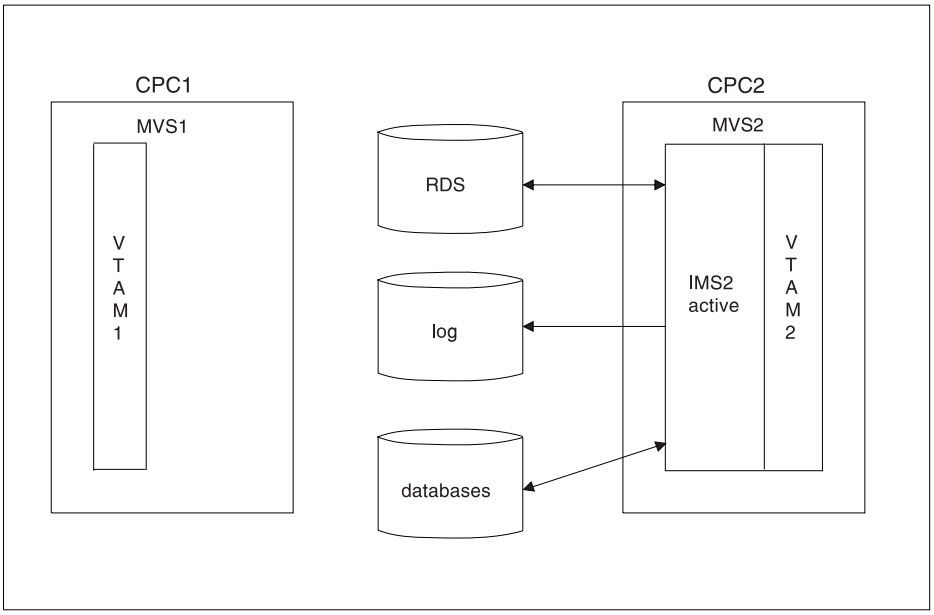


Figure 27. XRF Termination Phase

Normal Shutdown

In a normal shutdown (`/CHECKPOINT FREEZE`) of the XRF complex, IMS checkpoint processing occurs. When this has completed, IMS writes a X'06' log record and the alternate IMS terminates automatically. The active IMS tells the availability manager that processing is terminating with I/O prevention.

Planned Takeover

A `SWITCH SYSTEM` command is entered on the active IMS, a quiesce of dependent region processing is done on the active IMS, and a X'06' log

record is written indicating that the alternate subsystem is to take over processing. The active IMS communicates to the availability manager that a normal takeover is occurring.

Abnormal Termination
In an abnormal termination of the active subsystem, an attempt is made to purge the log buffer and write a X'06' log record. The active IMS terminates from the availability manager abnormally.

The two types of termination of the alternate IMS are:

Normal Shutdown
The operator can stop the alternate IMS by entering `/STOP BACKUP` on the alternate IMS or `/CHECKPOINT FREEZE` or `/CHECKPOINT DUMPQ` on the active IMS. In both cases, the alternate IMS leaves the availability manager and

Abnormal Termination
The IMS ESTAE exit is entered and the alternate subsystem terminates abnormally from the availability manager.

Cycle of XRF Phases

The five phases of XRF form a cycle. When an operator brings up a new alternate IMS, the two systems in the complex have exactly reversed their roles from the initialization phase. The left side of Figure 28 labeled “INITIALIZATION PHASE” appeared earlier in this chapter. Compare it with the diagram on the right that shows the XRF complex after a new alternate IMS is restarted.

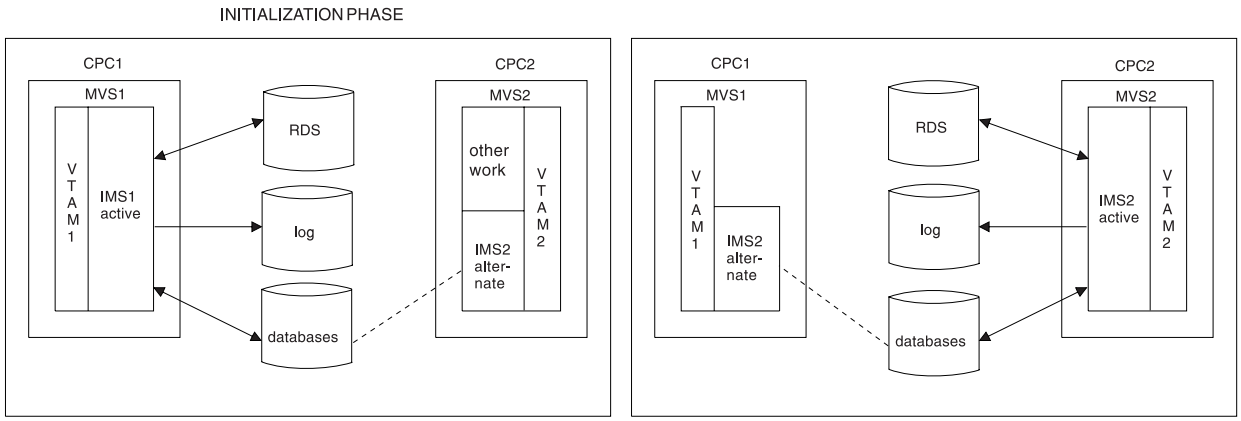


Figure 28. XRF Cycle

Organization of XRF Complexes

The XRF complex can be organized in several ways:

- One XRF complex with one CPC
- One XRF complex with two CPCs
- Two XRF complexes with three CPCs
- Two XRF complexes with four CPCs

One XRF Complex with One CPC

An XRF complex can be set up on one CPC with two IMS subsystems defined. However, the full benefit of XRF IMS is only achieved when IMS as well as MVS,

VTAM, and the CPC are replicated. When only one CPC is used, only IMS has an alternate system. Failures in MVS, VTAM, and the CPC affect both subsystems. This configuration consists of:

- One CPC, running MVS and VTAM
- Two IMS subsystems, the active and the alternate IMS subsystems
- IMS system logs shared by the active and the alternate IMS subsystems
- Databases shared by the active and the alternate IMS subsystems
- Remote terminals that use SNA protocol and that can communicate with either of the IMS subsystems
- NCP running in each 37x5 Communication Controller to which the SNA terminals are attached

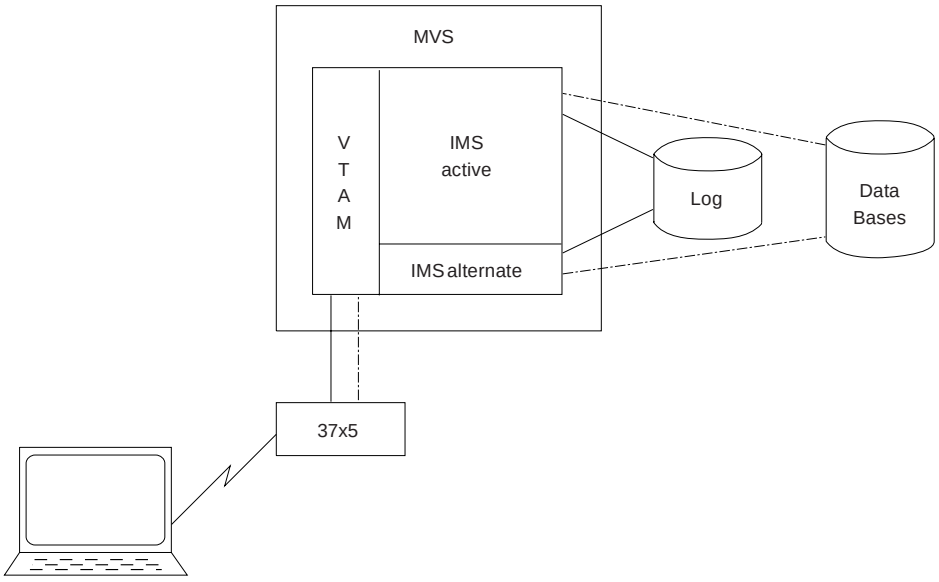


Figure 29. One XRF Complex in One CPC

This arrangement is recommended only as a test environment, and then only if the CPC is fast enough and the system has enough real and virtual storage to support XRF.

Processing in a One-CPC XRF Complex

In an XRF complex that runs on one CPC, XRF processing differs little from XRF running on two CPCs. One important difference is the role of the availability manager during takeover. In a 2-CPC XRF complex, the alternate IMS issues a `RESERVE` command to the IMS log to prevent the active IMS from accessing the databases. In a 1-CPC complex, this action does not prevent the active IMS from changing the databases. I/O prevention quiesces the outstanding I/O requests to the system log, as well as to the database data sets. When I/O prevention is complete, the operator can respond “GO” to AVM006A to complete the takeover. If the availability manager cannot perform I/O prevention, the alternate IMS waits for termination of the failing active IMS before completing the takeover.

Planning Considerations in a One-CPC Complex

In a 1-CPC XRF complex, all messages from the availability manager appear at the one MVS console as if two MVS operating systems were in the complex.

If the availability manager fails to perform I/O prevention at takeover, the IMS operator cannot reset the CPC to prevent the active IMS from writing to the databases. This action terminates the XRF complex. In a 1-CPC complex, the operator should terminate IMS and ensure that all the address spaces in the active IMS terminate before assuming that I/O prevention is complete.

One XRF Complex with Two CPCs

This configuration, shown in Figure 30, is the most typical and is used for the information in this chapter.

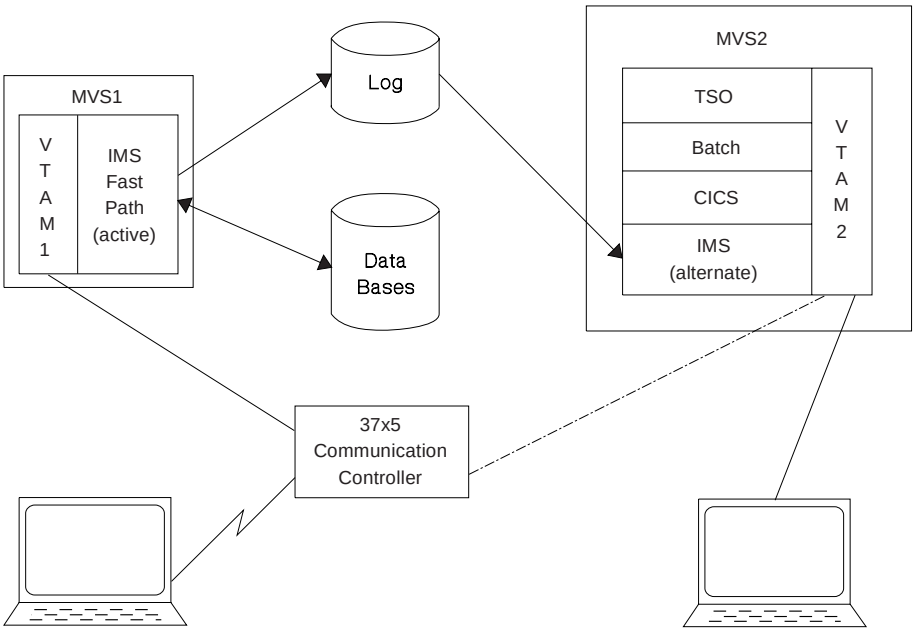


Figure 30. One XRF Complex in Two CPCs

One XRF Complex with Two CPCs and a Non-XRF IMS

This configuration, illustrated in Figure 31 on page 197, requires:

- Three CPCs, MVSs, and VTAMs
- Three IMS subsystems—two active and one alternate
- Databases shared by both active IMS subsystems and the alternate IMS subsystem
- IMS system logs for each active IMS subsystem and one shared system log with the alternate IMS subsystem
- Remote terminals using SNA protocol and communicating with one or both of the active IMS subsystems
- NCP running in the 37x5 Communication Controller attached to the CPCs with XRF

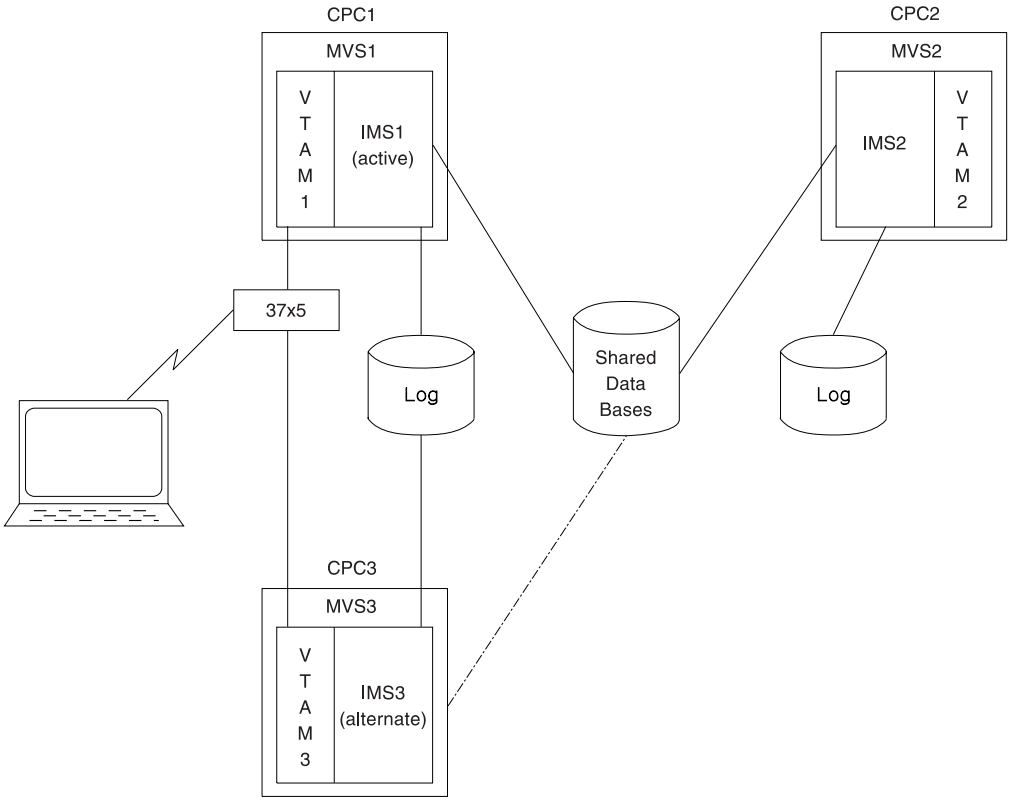


Figure 31. One XRF Complex in Two CPCs and a Non-XRF IMS

Two XRF Complexes with Three CPCs

You can have two XRF complexes running on three CPCs as shown in Figure 32 on page 198, providing adequate real and virtual storage is available on the CPC that runs the two alternate subsystems. This configuration provides an advantage over two XRF complexes in four CPCs; your alternate IMS systems both run in a single CPC, eliminating the need for one VTAM, one MVS, and one CPC. This configuration consists of:

- Three CPCs, MVSs, and VTAMs
- Four IMS subsystems—two actives and two alternates
- IMS system logs for each active subsystem and shared by its alternate subsystem
- Databases shared by the active subsystem and alternate subsystem in each complex
- Remote terminals using SNA protocol and communicating with one or both of the active IMS subsystems
- NCP running in each 37x5 Communication Controller attached to the CPCs

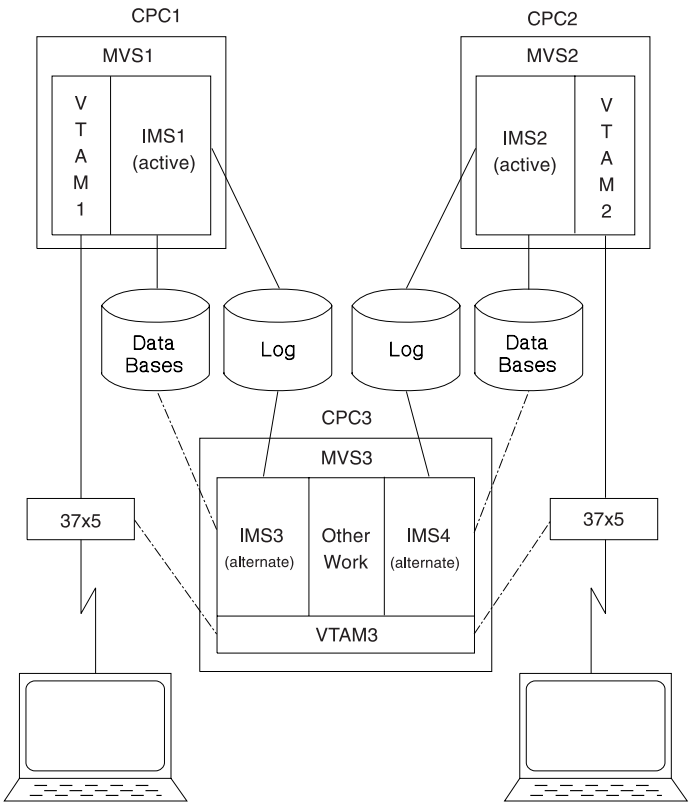


Figure 32. Two XRF Complexes in Three CPCs

Paths must exist to both the active and alternate subsystems for users of each subsystem. Each database must have shared access, either online or switchable at takeover, to its associated active and alternate CPC.

Two XRF Complexes with Four CPCs

Two XRF complexes that include data sharing must have IRLMs present in all CPCs. This configuration consists of:

- Four CPCs, MVSs, VTAMs, and IRLMs
- Two IMS active subsystems, each on a preferred CPC (IMS1 and IMS2)
- Two IMS alternate subsystems, each on a preferred CPC (IMS3 and IMS4)
- Two IMS system logs shared by the active and alternate subsystems
- Databases shared by the active and alternate subsystems
- Remote terminals using SNA protocol and communicating with one or both of the active IMS subsystems
- NCP running in each 37x5 Communication Controller attached to the CPCs

Figure 33 on page 199 shows two XRF complexes in four CPCs.

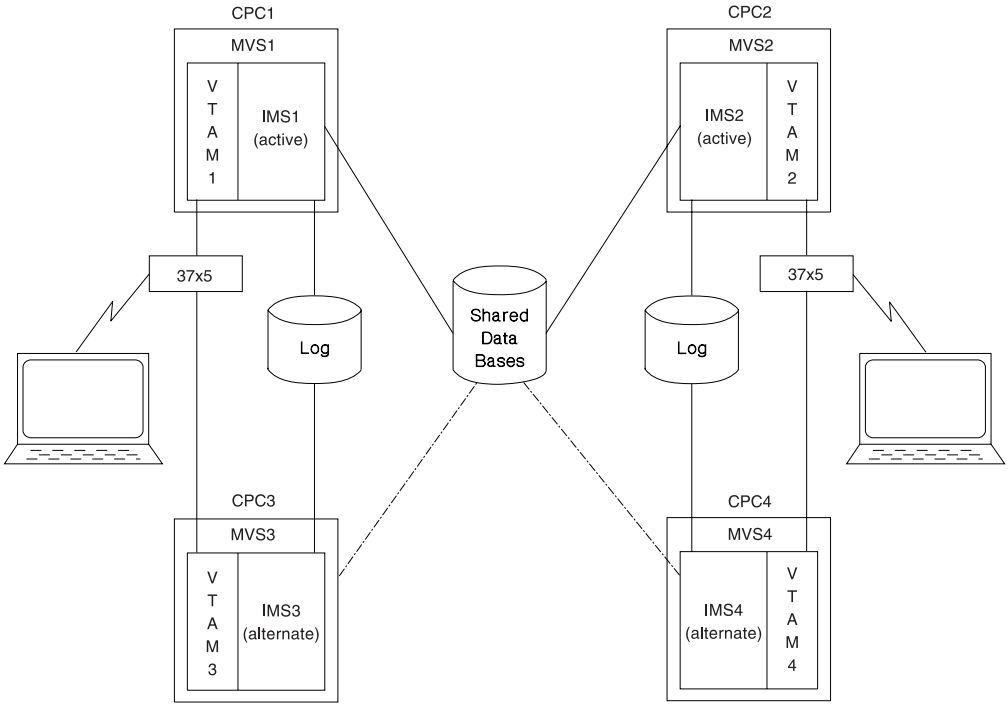


Figure 33. Two XRF Complexes in Four CPCs

Common access paths are in place on all IMS subsystems for RECON data sets and all shared databases.

Each IMS subsystem has an IRLM that is reconfigured during a takeover. The four IRLM sessions for this configuration are:

- Primary session between the IRLMs for IMS1 and IMS2
- Secondary session between the IRLMs for IMS1 and IMS3
- Secondary session between the IRLMs for IMS2 and IMS4
- Secondary session between the IRLMs for IMS3 and IMS4

Figure 34 on page 200 shows the IRLMs and sessions that the two data-sharing complexes need.

XRF Complex Organization

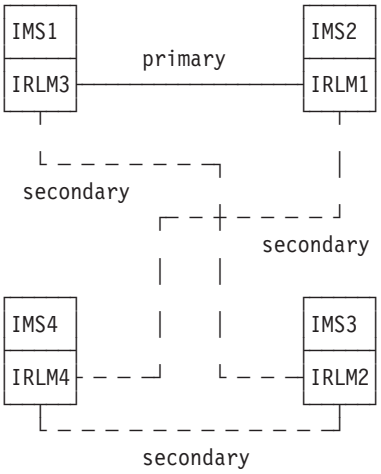


Figure 34. IRLM Sessions in an XRF Data Sharing Complex

IRLMs can be defined using only the APPL2 and APPL3 parameters. A secondary session is promoted to primary when two active IMS subsystems identify to two different IRLMs having a secondary session. The promoted session remains primary until a takeover forces reconfiguration or until a valid data sharing situation arises with a third IRLM. If a primary session is disrupted through session failure or loss of one of the IRLMs, that session can be reestablished and automatically becomes primary if another primary session has not already developed.

The two sides of a session must be defined equally, and each session must be unique; that is, two APPL2 sessions cannot be defined. The APPLS parameter overrides the automatic session determination until a takeover occurs.

When a primary session is demoted to a secondary session after a takeover occurs, no operator intervention is required.

Planning an XRF Complex

You must make some planning decisions about the takeover, such as some of the conditions that force a takeover, the amount of operator involvement, what terminal sessions recover automatically at takeover, and what methods the alternate IMS uses to learn of problems in the active IMS.

This section provides the following information:

Limitations of XRF

A takeover is not always performed when a failure occurs at an IMS installation. For example, XRF does not address the outages caused by failures of:

- A channel or link failure that causes a break in communication between the CPC and the network communication controllers
- Failures in the telecommunication network, such as controllers, NCP, lines, and terminals
- Intersystem failures, such as those caused by JES or CTCs
- Some failures in application programs
- Loss of or damage to the IMS databases or log
- A power failure that affects both CPCs in the complex
- Some operator errors

Related Reading: For information on the MVS Restart Manager (ARM) in an XRF complex, see “MVS Automatic Restart Manager (ARM) in an XRF Complex” on page 211.

Non-XRF Workload on the Alternate IMS

Processing the takeover significantly increases the work of the alternate IMS. Your installation must prepare to have the processing of the non-XRF workload on the alternate IMS slow down or cease. If the work can be swapped, MVS might swap it out temporarily. When MVS swaps the workload back in, it runs in a degraded mode, depending on how demanding the IMS workload is on the CPC. If you cancel the jobs on the alternate IMS, issue the `CANCEL` command for one job at a time after the takeover is complete. If you don’t wait for the takeover to complete, the takeover processing slows down because `CANCEL` command processing takes priority over takeover processing.

The non-XRF work that remains on the alternate IMS competes for resources with XRF-related work:

- At takeover, the work that cannot be swapped competes with takeover processing. SRM tries to give the alternate IMS the resources to take over the workload from the active IMS. SRM speeds up its analysis of the alternate IMS’s need for storage. The takeover might cause high paging activity and slower response time for the work that cannot be swapped. However, processing does continue.
- After takeover, the work that cannot be swapped competes with the IMS interactive workload. Response time for the non-XRF users depends on how you code the installation performance specifications (IPS) settings and on the demand for real storage by the work that cannot be swapped.

If you run TSO on the alternate system, it should have a lower priority than IMS. At takeover, MVS swaps out the TSO user address spaces if it needs the real storage. These address spaces remain swapped out until the new active IMS is stable; the duration depends on the installation profile. The alternate IMS should have IPS settings and SRM parameters that are the same as those for the active IMS, in order to favor IMS. When the takeover occurs, the IMS workload is favored.

Using the Intersubsystem Communication Link

XRF can make effective use of an intersubsystem communication link (ISC) session between the active IMS and the alternate IMS. The session is useful because:

- As surveillance, it alerts the alternate IMS to failures in the active IMS.

- During the synchronization phase, it allows the active IMS to take a SNAPQ checkpoint without operator intervention.

The ISC link is a normal VTAM APPL-to-APPL session and can be a route through any of the following:

- A channel-to-channel protocol (CTC or 3088)
- A 37x5 Communication Controller with channel paths to both systems
- Separate 37x5 Communication Controllers through a teleprocessing (TP) line

Figure 35 illustrates the three kinds of ISC links. Generally, the more hardware units there are on the route, the slower the communication and the greater the chance for failure. Choose the most reliable routes you can for surveillance and other communication.

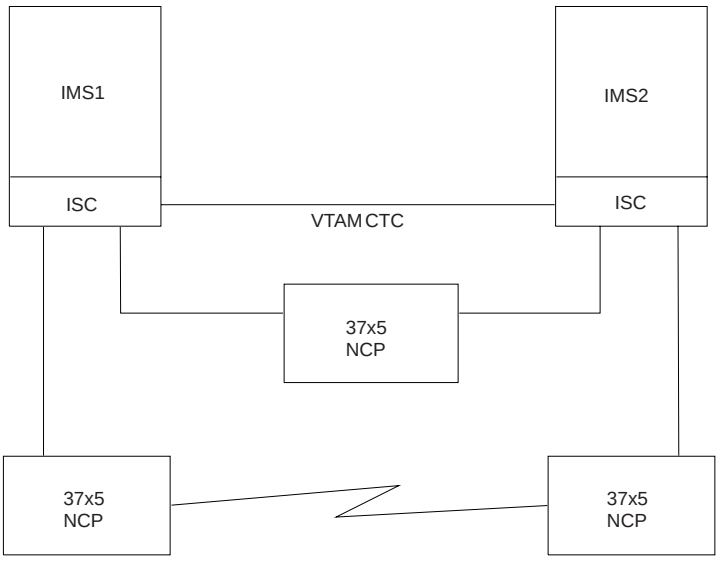


Figure 35. ISC Link Options

Terminals in an XRF Complex

XRF supports all of the terminals that presently serve IMS users. It gives this support with a minimum of effort from you. XRF IMS requires that you specify keywords on four or five system definition macro statements or ETO logon descriptors. Unless you want to change the switching priority of a terminal from the defaults or you want to change the terminal type sessions, you need not change any system definition macros.

XRF IMS gives the terminals at your installation the best service that the terminal's characteristics allow. This section describes these characteristics.

For the purposes of this explanation, the word "terminal" is used to describe a component of a terminal system, including a programmable controller and its attached operator terminals, printers, and auxiliary control units. It also describes remote subsystems.

XRF provides three classes of service to terminals: Class 1, Class 2, and Class 3.

Class 1 Terminals

For you to take full advantage of XRF, most of the terminals in your complex should be Class 1 terminals. Class 1 terminals are those terminals that:

- Use SNA protocol
- Are controlled by a VTAM and NCP that support XRF
- Are connected to a 37x5 Communication Controller
- Can be defined on the UNITYPE keyword on the TYPE macro in the IMS System generation as one of the following:
  - SLUTYPE1
  - SLUTYPE2
  - SLUTYPE4
  - SLUTYPEP
  - FINANCE

Class 1 terminals have backup sessions on the alternate IMS. These sessions are established during the tracking phase, when a session on the active IMS is initiated. They are switched automatically after a takeover without loss of the session.

For Class 1 terminals, you can specify the priority that controls the order in which IMS tells NCP through VTAM to switch sessions at takeover. Use the BACKUP= parameter on system definition macros or ETO logon descriptors to specify this priority. Setting a high priority for a terminal does not guarantee that a user will receive earlier session recovery at takeover; because of the competition for resources and transmission speed across lines, recovery is sometimes unpredictable.

Generally, the degree of transparency at takeover depends on the:

- Type of terminal
- Activity on the terminal at a takeover
- Session recovery priority assigned to the terminal

For example, a FINANCE terminal with no output or input in progress at takeover and a BACKUP=7 priority has the most transparent recovery.

From the terminal user's point of view, only one session with IMS exists. From the NCP point of view, two host sessions exist. The actual messages are only passed between the logical unit (LU) and the active IMS. The alternate IMS tracks the status of the message activity by monitoring the IMS log.

When the active IMS establishes or terminates a terminal session with Class 1 terminals, the alternate IMS establishes or terminates a backup session. When the active IMS fails, the alternate IMS takes over the terminal session without losing control from the viewpoint of the LU by changing the mode of the pre-established backup session from BACKUP to ACTIVE. When the session is switched, the NCP sends the alternate IMS its view of the terminal's status. IMS compares this with its own record of status to decide what, recovery action to take, if any. When the REVERIFY operand is used with RACF, the user must sign on again.

Class 2 Terminals

Class 2 terminals do not have backup session support on the alternate IMS, but are restarted automatically after takeover. The interface to the terminal, including

recovery procedures after session reestablishment on the alternate IMS, is exactly the same as in the prior version of IMS, except for the automatic session re-establishment at system takeover.

When a Class 2 terminal session is established on the active IMS, the alternate IMS tracks the session initiation or termination using the log records. When the active IMS terminates abnormally, the alternate IMS tries to establish a new session with the network resources that were active before the failure.

Class 2 terminals receive good support from XRF. In fact, at takeover, IMS might be able to reestablish service on some of your Class 2 terminals before all of your Class 1 terminal sessions are switched. Using a BACKUP= parameter on system definition macros or ETO logon descriptors, you can set the priority that controls the order in which IMS reestablishes service to Class 2 terminals at takeover.

When IMS tries to reestablish a session with a terminal to which there is no available path, the attempt fails. Before IMS can reestablish a session on a locally attached terminal, the operator must switch the terminals to the new active IMS. If the complex has many of these kinds of terminals, you might consider giving the operator control at takeover before the takeover actually proceeds. At this time, the operator can perform the switch, allowing a faster recovery for sessions on these terminals. You give the operator this control in the AUTO parameter in member DFSHSBxx in IMS.PROCLIB.

Although IMS tries to establish new sessions for the Class 2 terminals, if the owner of the terminals or the link to the owner fails, the attempt to establish the new session also fails.

Terminals that qualify as Class 2 are:

- Non-switched terminals controlled by BTAM
- Multiple Systems Coupling (MSC) and ISC subsystems that communicate with IMS XRF through VTAM or bisynchronous lines
- Terminals on leased lines controlled by BTAM and connected to a 37x5 Communication Controller with multi-system line access (MSLA)
- Bisynchronous 3270 terminals attached to a 37x5 Communication Controller
- Spool line groups on shared DASD, locally attached to both CPCs
- Non-SNA 3270 terminals controlled by VTAM
- Devices controlled by the Network Terminal Option (NTO) licensed program
- Locally attached devices
- Terminals that do not use the SNA protocol
- Terminals that qualify as Class 1 terminals, except:
  - The terminals are not controlled by a 37x5 network communication controller
  - The terminals are not controlled by an NCP or VTAM that supports XRF or defined BACKUP=(n,NO) on the system definition macro or ETO logon descriptor

If you have an application terminal that communicates with a programmable controller in an XRF complex, you might need to add code to the communications portion to achieve the highest level of transparency available. Additionally, for the ISC terminal, you must define a session for the link that connects the ISC terminal to XRF IMS.

Related Reading: For information on what code you need to add to the programs, see *IMS/ESA Customization Guide*.

Class 3 Terminals

Class 3 terminals do not have backup session support on the alternate IMS. Their terminal sessions must be restarted manually on the new active IMS after takeover, either by the MTO or by user logon. It is recommended that Class 3 terminals be switched in the same manner that these terminals were switched in prior releases of IMS when an operator brought up a different CPC after a failure.

Terminals that are eligible for Class 3 include all the terminals eligible for Class 3 service (such as all LU 6.2 devices), and Class 1 and Class 2 terminals that have indicated no switching should be done. This is done by specifying BACKUP=NO on the system definition macro or ETO logon descriptor.

Users at Class 3 terminals use the existing procedures to communicate with the active IMS after takeover. If their terminals are not physically connected to both IMS subsystems, reestablishment of sessions might not be possible.

Specifying Terminal Support

All terminals that IMS supports can be used in an XRF complex. The terminal characteristics determine the type of support a terminal can have in an XRF complex, with Class 1 having the best support and Class 3 having no support.

LU 6.2 devices are always Class 3. ETO terminals and users are eligible for any level of support.

Related Reading: For more information on LU 6.2 and ETO in XRF, see *IMS/ESA Administration Guide: Transaction Manager*.

Table 30 summarizes the keywords used to change the priority or terminal type of various devices.

Table 30. Defining Class of XRF Service

Device	Class	BACKUP Keyword on TERMINAL Macro or ETO Logon Descriptor
SLUTYPE1 (3286,...) SLUTYPE2 (3278,3279,...) SLUTYPE4 (6670)	Eligible for CLASS 1,2,3	For Class 1 service, BACKUP=(1-7,YES)
LUTYPE6 (ISC) MSC (VTAM)	Eligible for CLASS 2,3	For Class 2 service, BACKUP=(1-7,YES)
MSC (BSC) NTO (leased line) 3270 (VTAM) Spool line groups BTAM (non-switched line) 3270 (BTAM) Locally attached devices		For Class 3 service, BACKUP=NO

The BACKUP keyword on several macros in the system definition statement specifies the support a terminal receives. Your installation determines which terminals have backup sessions established and the switching priority of the terminals. Level 1 is the lowest priority, and level 7 is the highest. Although VTAM

requests are priority ordered by IMS, the requests can be completed in any order because of unpredictable factors in the network, such as pacing, line speeds, and congestion.

If BACKUP=(1-7,YES), this terminal is automatically taken over without losing session control after takeover, and a backup session is established during tracking. This constitutes Class 1 support as long as the terminal is connected to a 37x5 and XRF-capable VTAM and NCP.

The maximum number of terminal backup sessions that can be established on the alternate IMS is specified in the NCP BUILD statement with the BACKUP=n option. If the (n+1) Class 1 terminal user tries to logon to the active IMS, this logon is denied. Carefully determine what the maximum number of backup sessions should be on the alternate IMS, and specify an appropriate number.

If BACKUP=(1-7,NO), this terminal is automatically restarted after takeover, but no BACKUP session is established. This forces Class 2 support.

For both Class 1 and Class 2 terminals, the default switching priority is 4.

If BACKUP=NO, this terminal must be restarted manually after takeover. This forces Class 3 support.

How Takeover Affects a Terminal User

A user's awareness of a takeover depends on the class of terminal, the priority with which IMS requests session recovery, the access method that controls the terminal, and the status of the input or output message at the time of failure. The main network objective is to recover service on the terminals as soon as possible. As sessions on Class 1 and Class 2 terminals recover, users might notice a pause in operations. If a user on a Class 1 terminal has just entered a message, the alternate IMS might request that the user reenter the message or issue message DFS3861I. In both cases, the user should clear the screen and reenter the last message.

Related Reading: For more information on how to ensure that users of FINANCE and SLUTYPEP terminals receive this message, see *IMS/ESA Customization Guide*.

IMS notifies most Class 2 terminal users of the recovered sessions by issuing message DFS2469W, DFS3649, or DFS3650, indicating that the session has been reconnected. FINANCE, SLUTYPEP, and ISC terminals do not receive these messages.

Related Reading: For more information on these messages, see *IMS/ESA Messages and Codes*.

The IMS operator uses existing IMS procedures to recover service on terminals that fail at a takeover.

Takeover for Class 1 Terminal Users (except SLUTYPE2 Users)

If the user is between transactions, has received a reply to the last data entry, and has not yet begun the next transaction, no indication exists that a takeover has occurred and that the terminal has been switched. The user might notice that a takeover has occurred if a failure occurs during input, output, or when transactions are in-flight.

- **Input**

  Queued Recoverable
  If the input message of a recoverable transaction has been queued, it is automatically scheduled after takeover and the user is not aware of the terminal switch.

  Not Yet Queued Recoverable
  If the input message of a recoverable transaction had not been queued at the time of the failure, the message is lost. This includes Fast Path transactions that had not reached a sync point.

  Conversational
  If the transaction was conversational, the alternate IMS sends an Exception Response (SSENSE=X'0826', USENSE=X'0F15') message. DFS3861I is also sent if the terminal can receive system messages. The RTO can issue `/HOLD` and `/RELEASE` commands for the conversation and receive the first page of the final output again.

  Non-conversational
  If the transaction was non-conversational and the last output message was not MFS, the Exception Response and system message are sent (see Conversational, above), and the RTO resumes with a new transaction.

  Queued Nonrecoverable
  If the transaction is nonrecoverable and has been queued, the message is lost.

  Not Yet Queued Nonrecoverable
  If the transaction is nonrecoverable and has not been queued, the message is lost. The alternate IMS sends an Exception Response (SSENSE=X'0826', USENSE=X'0F15') and message DFS3861I.

- **In-Flight**

  Recoverable
  If the transaction is being processed at the time of failure, IMS backs out the updates, reschedules the transaction, and sends the appropriate output message after completion of the transaction. The takeover is transparent to the user only if there is no delay in scheduling the transaction. Otherwise, the terminal user experiences a response-time delay. The user does not need to reenter any data.

  Nonrecoverable
  If the transaction is nonrecoverable or is a Fast Path transaction that has not reached sync point, the transaction is lost. The Exception Response (SSENSE=X'0826', USENSE=X'0F15') and message DFS3861I are sent.

- **Output**

  Recoverable
  If the output message at the time of failure is recoverable, the alternate IMS continues to send the message from the next segment, and the terminal user is not aware of the takeover.

  Nonrecoverable
  If the output message at the time of failure is nonrecoverable, the alternate IMS resets the current message and sends message DFS3861I if the terminal can receive system messages.

Unknown Output

If the alternate IMS does not know the contents of the last output message (if the last output was a system message which is not logged), the alternate IMS cancels the current output message and sends message DFS3861I if the terminal can receive system messages.

Takeover for Class 1 SLUTYPE2 Terminal Users

Terminal users at SLUTYPE2 devices are always aware of takeover activity. If the terminals are defined to receive system messages, they receive message DFS3861I text after the takeover. I/O activity at the time of failure is always reset.

- **Input**

  Queued Recoverable

  If the input message of a recoverable transaction is queued, it is automatically scheduled after takeover. The terminal user must press the PA1 or PA2 key to receive the output if the terminal is defined to receive DFS3861I messages.

  Not Queued Recoverable

  If the input message of a recoverable transaction is not queued at the time of the failure, the message is lost. This includes Fast Path transactions that had not reached a sync point.

  1. Conversational

     If the transaction is conversational, the terminal user can receive the first page of the final output by entering `/HOLD` and `/RELEASE` commands.

  2. Non-conversational

     If the transaction is non-conversational, the terminal user must begin a new transaction. A `/FORMAT` command might be needed.

  Nonrecoverable

  If any nonrecoverable transaction or the resultant reply is in the failed system at the time of failure, the transaction or message is lost.

- **In-Flight**

  Recoverable

  If the transaction is being processed at the time of failure, IMS backs out the updates and reschedules the transaction. The terminal user must press the PA1 or PA2 key to receive the output message. The user might experience a response-time delay if the transaction is not able to be scheduled immediately after takeover. The user does not need to reenter the transaction.

  Nonrecoverable

  If the transaction is nonrecoverable or is a Fast Path transaction that has not reached sync point, the transaction is lost.

- **Output**

  MFS Logical Paging

  If the terminal user is paging through logical pages, the user must press the PA1 or PA2 key to get the first screen of the final set of logical pages that are being reviewed.

  Single MFS Page or Non-MFS Output—Not Dequeued

  If the failure occurred between the time IMS sent the output and the time IMS received an acknowledgement of receipt (the message is not dequeued), the final output is sent to the terminal again.

Single MFS Page or Non-MFS Output—Dequeued
If the failure occurs after the output is sent and the acknowledgement is received, the user must begin the transaction again. A /FORMAT command might be needed.

MFS Multipage Output without Operator Logical Paging—Last Page Send and Not Dequeued
If the failure occurs between the time IMS sent the output and the time IMS received an acknowledgement (the message is not dequeued), the first page of the final output is sent again.

Last Page Sent—Dequeued
If the failure occurs after the output is sent and the acknowledgement is received, the user must begin the transaction again. A /FORMAT command might be needed.

Last Page Not Sent
If the last page is not sent at the time of failure, IMS sends the first page of the output again. The terminal user pages with the PA1 key.

Takeover for Class 2 Terminal Users

Class 2 terminal users have the same support that they have in a non-XRF environment after an emergency restart. The difference is that the LOGON sequence has been handled automatically by takeover. Users at FINANCE, SLU P, and ISC terminals are signed on automatically at takeover. Other ETO users might be required to signon again after takeover.

3270 VTAM
After takeover, IMS attempts to establish a new session on the alternate IMS.

LUTYPE6 (ISC)
After takeover, IMS attempts to establish a new ISC session on the alternate IMS.

MSC (VTAM and BSC)
After takeover, IMS attempts to reestablish all VTAM and BTAM physical links that were active at the time of failure.

NTO After takeover, IMS attempts to establish a new session on the alternate IMS.

BTAM non-switched terminals
After takeover, IMS attempts to restart all BTAM terminals that were active at the time of failure and that were connected to a 37x5 with the MSLA feature.

Locally attached devices
These devices must be switched manually.

LU 6.2 devices
Conversations in process must be reallocated.

Remote devices
VTAM terminals that are twin tailed or that are local-to-local NCP connected are switched automatically. If the terminal is not physically connected to the new active IMS, the switch fails.

If the owning VTAM fails, or in the case of a BTAM terminal, the operator must switch the terminals to the new active IMS. Device switching can also be done using a 2914/3814.

Spool line groups
After takeover, IMS opens a new spool data set.

How VTAM Ownership Affects Terminal Switching

Terminals can be owned by a VTAM residing in the active CPC, in the alternate CPC, or in a CPC of a communication management configuration (CMC). Having your terminals owned by a host outside the XRF complex offers the most stable ownership, because the host is not affected by takeover, and the takeover is not affected by the CMC activities.

Table 31 summarizes the terminal's session status, logon capabilities, and logoff capabilities after a takeover. It assumes that VTAM in the active system is no longer functioning.

Table 31. Terminal Capabilities of Owning VTAM during and after a Takeover

	Active System	Alternate System	CMC or Other Host System
Terminal class	1 2 3	1 2 3	1 2 3
Session continue	Yes No No	Yes No No	Yes No No
LOGOFF enabled	Yes N/A No	Yes N/A No	Yes N/A N/A
LOGOFF enabled[1]	Yes Yes Yes	Yes Yes Yes	N/A N/A N/A
LOGON enabled	No No No	Yes Yes Yes	Yes Yes Yes
LOGON enabled[1]	Yes Yes Yes	Yes Yes Yes	N/A N/A N/A

Note:

1. After another system acquires terminal ownership

VTAM USERVAR Table Definition

The following examples show the relationship between the IMS system definition parameters and the VTAM interpret USERVAR table parameters used with XRF.

Related Reading:

- For more information on the USERVAR table, see "Terminal Logon in the XRF Complex" on page 177.
- For more information on IMS system definition parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

You should perform the following tasks:

- You must specify the APPLIDs and passwords of both systems in the APPLID and PASSWORD parameters of the COMM macro at IMS Generation. For example:

```
IMSCTRL HSB=YES
COMM APPLID=(IMS1,IMS2),PASSWORD=(IMSP1,IMSP2)
```

- Before starting the IMS system, you must set up the DFSHSBxx member of the IMS-PROCLIB specifying the Generic Application ID in the USERVAR parameter. For example:

```
USERVAR=IMS
```

- The Generic Application ID must be registered into the VTAM interpret table as a user variable. For example:

```
LOGCHAR APPLID=(USERVAR,IMS),SEQUENCE='IMS'
```

- APPLIDs specified in APPLID parameter of the COMM macro must be defined as either SPO (secondary programmed operator) or PPO (primary programmed operator) in the VTAM definition library. For example:

```
      VBUILD TYPE=APPL
IMS1  APPL   AUTH=(ACQ,SPO),PRTCT=IMSP1,HAVAIL=YES
IMS2  APPL   AUTH=(ACQ,SPO),PRTCT=IMSP2,HAVAIL=YES
```

BTAM Ownership of Terminals

Although VTAM is the primary access method in XRF, Class 2 terminals controlled by BTAM receive good support from XRF, provided they are:

- Nonswitched
- Connected to the 37x5 Communication Controller with the multi-subchannel line access (MSLA) feature

Performance Considerations

When you evaluate a configuration with XRF capability, you should distinguish between general availability criteria and response-time objectives. By carefully planning the configuration, you can provide alternative access paths and reduce the single point-of-failure occurrences. If you consider the total system response characteristics, you evaluate the response for each system when active. Your initial parameters for IMS, page fixing, and region utilization are sized to the CPC and the expected workload.

Response-Time Objectives

To account for special situations, you might need to adjust the objectives you have established for the workload on either CPC. For example, exclude from your statistics the situation in which a successful takeover occurs from a planned outage. If you set any objectives for critical transactions, remember that takeover processing is normally followed by a period when response is stabilizing. An end user might experience a slight delay.

Capacity Planning

When planning the virtual storage requirements in an XRF complex, the key question is: how much resource is required to give acceptable support to a given IMS user set? Careful estimates should be made if more than one IMS alternate subsystem is to be run on the same alternate CPC.

MVS Planning Considerations

MVS systems without XRF can coexist with MVS XRF systems in a single installation. In particular, MVS XRF and non-XRF systems can participate in a single global resource serialization (GRS) ring and in a single JES2 or JES3 complex. Even in one MVS system, an XRF IMS subsystem can coexist with non-XRF jobs or subsystems, including another IMS.

MVS Automatic Restart Manager (ARM) in an XRF Complex

In an XRF environment, when the alternate IMS system has started the tracking phase, it issues an ARM associate. This ensures that the active IMS system is not automatically restarted by ARM after the backup takes over. If the active IMS were to automatically restart after the backup takes over, the IMS log (OLDS) and the message queue might be destroyed.

If IMS abends before it completes restart, the abended IMS de-registers from ARM and is not automatically restarted, because XRF tracking is considered to have completed restart for an XRF backup.

Global Resource Serialization Considerations

XRF does not require global resource serialization. However, if one CPC in an XRF complex is in a global resource serialization ring, the other CPC in the complex should be in the same ring. MVS with XRF and MVS without XRF can participate in a single global resource serialization ring.

XRF and the XRF takeover do not affect the normal procedures for global resource serialization. However, if your two systems are in a global resource serialization ring, you should be aware of the relationship between XRF and global resource serialization.

If the CPC or MVS fails in the active IMS, the alternate IMS subsystem requests a takeover. Messages from global resource serialization indicate that the ring is disrupted. When the system restarts (either automatically through the RESTART(YES) parameter in the GRSCNFxx member of SYS1.PARMLIB or through the operator issuing the `VARY GRS,RESTART` command), global resource serialization rebuilds the GRS ring that was disrupted. To release the resources owned by the failed system, the operator issues the `VARY GRS,PURGE` command. When the operator IPLs MVS and restarts IMS as the new alternate subsystem for the XRF complex, the system rejoins the ring.

If the CPC or MVS fails in the alternate system, processing on the active IMS continues normally. The operator should enter the `VARY GRS,PURGE` command to free the resources owned by the failed active IMS and relocate the alternate IMS on another CPC in the ring.

Include all IMS data set names in the global resource serialization SYSTEMS exclusion resource name lists (RNLs). Depending on the naming conventions at your installation, you might be able to include them with one generic entry. If you do not list the names in the SYSTEMS exclusion RNLs, global resource serialization serializes access to these data sets, a service that DBRC and the IRLM already provide the IMS data sets.

Do not include the DBRC RECON or the OLDS and WADS names in the RESERVE conversion RNL. Doing so results in physical reserves being sent to DASD when reserves are issued by IMS.

Backup CTCs are recommended, because they allow an installation to continue using GRS when a CTC fails. Without a backup CTC, all global ENQ requests remain suspended until the operator reestablishes communication between systems.

JES Considerations

MVS with XRF and MVS without XRF can both participate in a JES2 or JES3 complex. The active and alternate IMS subsystems can be in two JES3 global systems. If you use the services of JES3, include all IMS data sets and databases in the RESDSN statement. If you do not do so, and the JES3 global fails, the active IMS cannot allocate the data sets and databases until a JES3 global becomes available.

The IMS data sets must be available when JES3 is brought up.

RACF Considerations

Because the alternate IMS needs the security information in the RACF data sets at takeover, place the RACF data sets on DASD shared by the active and alternate IMS subsystems. To avoid single points of failure, use the RACF backup facility to keep a second copy of these data sets also on shared DASD.

The `REVERIFY` operand on the RACF `RDEFINE APPLDATA` command is nullified during a takeover. If your installation uses this operand, the users must sign on again after a takeover to identify the password to the new active IMS.

VTAM Planning Considerations

The following VTAM licensed programs must be Version 3 level or higher:

- Two VTAMs at an XRF complex.
- Any VTAM that owns a Class 1 terminal. (These VTAMs might or might not be in the XRF complex.)
- Any VTAM that serves application programs that open sessions with XRF IMS.
- Any VTAM that participates in opening backup sessions.
- Any VTAM that supports terminals that need the USERVAR table to log on to XRF IMS.

Determining Ownership of Class 1 Terminals

When determining where to place the ownership of the Class 1 terminals, consider what causes the least operator effort, the least disruption to Class 1 terminal users, and the least strain on the CPC. Before you make the decision, consider some of the implications of VTAM failures on ownership. First, review some information about VTAM that is not new, but has new meaning for XRF.

Ownership is significant when a terminal user logs on to IMS or logs off from IMS. Each VTAM has one system services control point (SSCP); it is responsible for session initiation and session termination for terminals that it owns. Therefore, if the owning VTAM fails, some functions of VTAM are no longer available. Service on the Class 1 terminals that are logged on to IMS does continue. But when a user of a Class 1 terminal within its domain tries to log on, VTAM cannot honor the request. Upon restarting VTAM, the network operator must also reestablish the ownership of the terminals. Alternatively, the network operator can transfer ownership to another VTAM.

At a takeover, the ownership of Class 1 terminals does not shift to the other XRF environment. If you assign ownership to the active VTAM and the active IMS fails, the Class 1 terminals switch to backup sessions, but the VTAM in the active system continues the ownership. When you return the failing active IMS to service as a new alternate IMS, the owning VTAM is in the alternate IMS.

In making the decision about terminal ownership, consider three options. The owner of the Class 1 terminals can be:

- The VTAM in the active system
- The VTAM in the alternate system
- A VTAM in a Communication Management Configuration (CMC)

You should assign the ownership of the Class 1 terminals to a third VTAM in a CMC. Having the owning VTAM in a CPC that is outside the XRF complex offers the following advantages:

- It reduces contention on the active system during normal operations.
- It reduces contention on the alternate system during the takeover.
- It prevents failures in the XRF complex from terminating the VTAM that owns the Class 1 terminals.
- It allows terminals that are connected to the NCP on SNA switched (that is, dial-up) lines to be Class 1 terminals.

If you do not have a CMC, assign the ownership of the Class 1 terminals to either VTAM in the complex, and do not change the ownership after a takeover unless you are restarting VTAM.

Related Reading: For more information on reestablishing or changing ownership of terminals, see *VTAM Operation*.

NCP Planning Considerations

The primary planning task for NCP is deciding how many Class 1 terminals you want to attach to each 37x5 Communication Controller. For each NCP, specify this number on the BACKUP operand on the BUILD definition statement. This action allows the NCP, through the SSP, to generate the appropriate number of control blocks for primary and backup sessions for Class 1 terminals.

The 37x5s connected to your Class 1 terminals need not be dedicated to the IMS work. They can also support other network traffic, including gateway. As you plan the storage requirements for the gateway 37x5 that supports Class 1 terminals, remember that this type of 37x5 requires more storage than a 37x5 without XRF terminals and without gateway.

Related Reading: For more information on NCP storage, see *Network Program Products Planning*.

Preparing the System for XRF

You define IMS to be part of an XRF complex during the IMS system definition. Do this by:

- Tailoring the IMS execution JCL (see page 214)
- Coding system definition macro statements to:
  - Establish XRF to be part of the generated system (see page 216)
  - Specify the VTAM application names and passwords (see page 216)
  - Define the IMS master terminal and secondary terminals (see page 216)
  - Customize individual terminals (see page 217)
  - Optionally declare priorities that affect the order in which terminals and MSC links are made available (see page 218)
- Using IMS.PROCLIB members to initialize system data sets (see page 218)

These three activities are described in the following sections.

Tailoring the IMS Execution JCL

Each CPC in an XRF complex has an IMS job running continuously. OS/390 START commands use two different procedures to initialize the active IMS and the alternate IMS. The two parameters in the EXEC statement for control regions that reflect the presence of an XRF complex are:

HSBID

Specified as HSBID=1 in one procedure and HSBID=2 in the other. HSBID=null deactivates XRF capability. If you bring up a third subsystem as an alternate system after a takeover, specify HSBID= the same as on the failed active system.

HSBMBR

Specified as a 2-character suffix that corresponds to a DFSHSBxx member included in IMS.PROCLIB. This member is where the control information and takeover criteria are specified.

Coding IMS System Definition Macro Statements

When you define the IMS system, you specify certain keywords on the macro statements that are in your existing generation deck.

Related Reading: For the syntax of the system definition macro statements, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Figure 36 shows the IMS system definition process and identifies the macro statements that have XRF-related keywords.

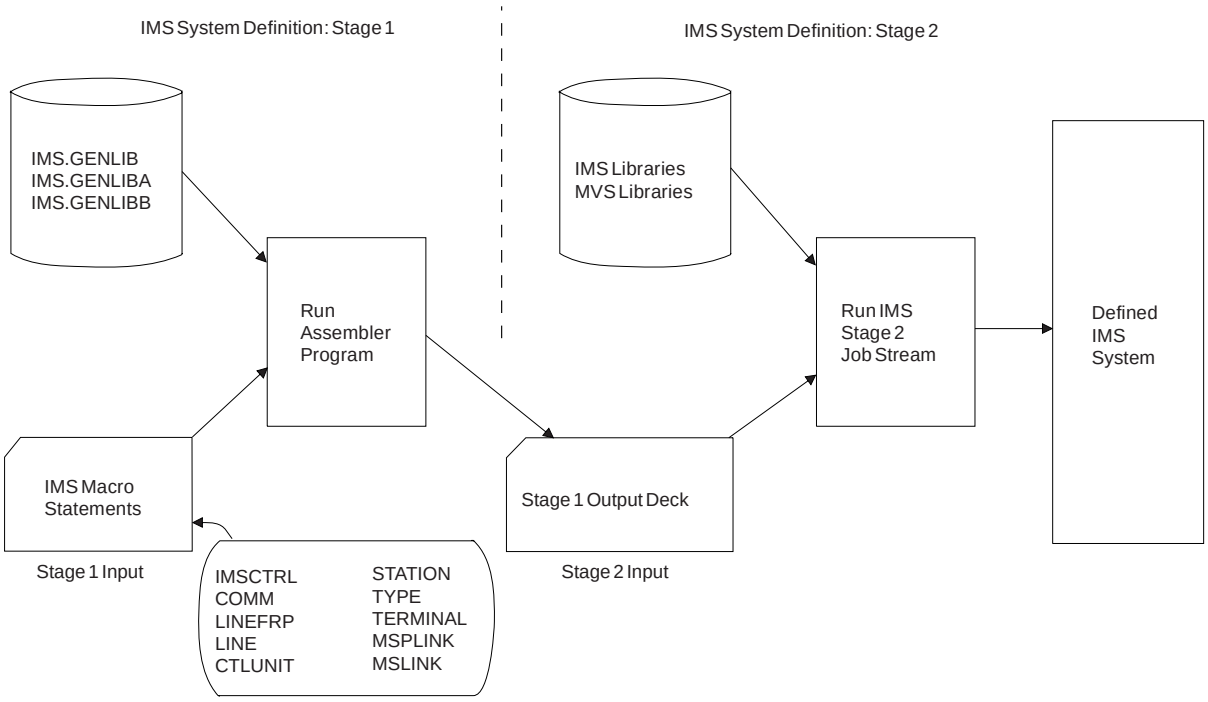


Figure 36. IMS System Definition Macros with XRF Keywords

XRF requires that you code the keywords that:

- Establish XRF support for IMS (see “Establishing XRF Support for IMS” on page 216).
- Provide two VTAM application names (see “Specifying the VTAM Application Names and the Passwords” on page 216).
- Name two passwords for ACB security (see “Specifying the VTAM Application Names and the Passwords” on page 216).
- Define your master and secondary IMS terminals (see “Defining the IMS Master and Secondary Terminals” on page 216).

You can use the defaults on other keywords that define what happens to terminals at a takeover. The defaults allow you to change IMS support at takeover and change the priority that IMS uses to recover sessions at takeover. If you allow XRF to use the defaults on the macros and ETO descriptors that define your terminals, you receive the best recovery IMS can give your terminals. The default priority for IMS to recover sessions at takeover is 4, with 7 being the highest priority.

Establishing XRF Support for IMS

To request that IMS be capable of running XRF, use the HSB keyword. On the HSB keyword on the IMSCTRL macro, specify YES. When you specify HSB=YES, ISC has duplexed blocks. Specify HSB=YES if you plan to run an alternate subsystem anytime before the next IMS system generation, because a system definition and an accompanying IMS cold start are required to change HSB from NO to YES.

Specifying the VTAM Application Names and the Passwords

To specify the name and password for IMS in the active and alternate subsystems, use the APPLID and PASSWORD keywords on the COMM macro. The APPLID keyword on the COMM macro defines the two VTAM application names (ACBs) that VTAM places in the USERVAR table. Only one application name is in the USERVAR table at any one time, the name that applies to the active IMS. At takeover, IMS tells VTAM to replace it with the other application name.

The PASSWORD keyword on the COMM macro defines two passwords that VTAM checks in the ACB.

For example, to specify VTAM application names of IMSNAME1 and IMSNAME2 with passwords of IMSP1 and IMSP2:

```
COMM APPLID=(IMSNAME1,IMSNAME2),PASSWORD=(IMSP1,IMSP2)
```

In this example, IMSNAME1 and IMSNAME2 must be the labels on the APPL definition statement that authorizes VTAM to support backup sessions for Class 1 terminals.

Defining the IMS Master and Secondary Terminals

Both the active and alternate IMS subsystems have IMS primary and secondary master terminals that are always active. To define them, specify the following XRF-related keywords on the TERMINAL macro:

- If VTAM controls your IMS master terminals, specify the NAME keyword.
- If BTAM controls your IMS master terminals and you cannot use the same address for both terminals, specify a second address on the LINE and ADDR keywords.

The NAME keyword on the TERMINAL macro defines a second node name for a VTAM master or secondary terminal. It is also used to define an IMS-managed ISC link between the active and alternate subsystems when both names match the APPLID keyword on the COMM macro.

The ADDR keyword on the LINE macro defines a second line address for a BTAM master or secondary terminal. When specified, only the master and the secondary master terminals can be in the line group.

The ADDR keyword on the TERMINAL macro defines a second terminal address for a remote BTAM master or secondary terminal. The corresponding LINE macro must not define any terminals other than the BTAM master or secondary terminal.

Customizing Individual Terminals

Regardless of the number of terminals in your complex, obtaining XRF support for those terminals does not require you to specify any additional keywords. By accepting the defaults, IMS determines the best recovery possible for your present terminals. For example, terminals that meet all of the following criteria are automatically defined as Class 1 terminals:

- Defined as SLUTYPE2 on the UNITYPE keyword
- Connected to 37x5 Communication Controllers
- Controlled by NCP and VTAM that support XRF

IMS automatically considers as Class 2 any terminal that is defined UNITYPE=SLUTYPE2 and is attached to a 37x5 Communications Controller.

Figure 37 shows the keywords you code to customize individual terminals. The BACKUP keyword on the TYPE, TERMINAL, LINEGRP, LINE, CTLUNIT, and

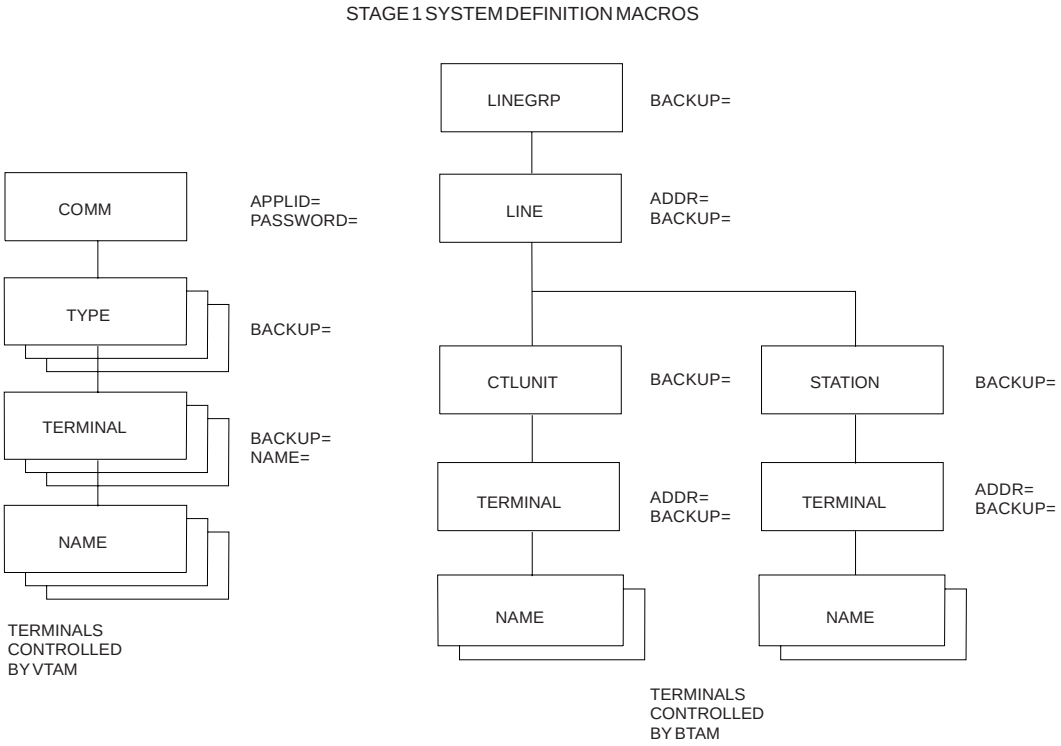


Figure 37. IMS System Definition Macro Keywords for Terminals

STATION macros and ETO descriptors specifies session recovery on Class 1 and Class 2 terminals and sets the priority in which IMS recovers service on these terminals. The default for the BACKUP keyword is (4,YES). This means terminals that qualify as Class 1 automatically have backup sessions at takeover with a switching priority of 4. Terminals that qualify as Class 2 terminals automatically have new sessions established at takeover with a priority of 4.

Specify BACKUP=NO if you have some reason to change Class 1 or Class 2 support to Class 3.

Use the BACKUP keyword to establish a priority other than 4 for recovery of service on your Class 1 or Class 2 terminals. Specify BACKUP=n, where n is a number between 1 (slowest recovery) and 7 (fastest recovery).

Although these priorities are used by IMS, because of internal VTAM processing, the requests for network connection might be completed in a different order.

To change Class 1 support to Class 2 support, code BACKUP=n,NO where n is the session recovery priority for the terminal.

Example 1: If you want all the BTAM-controlled terminals defined by a LINE macro to be Class 2 terminals with the session recovery priority of 7, but you do not want IMS to reconnect terminal TERMB at a takeover, specify:

```
LINE . . .BACKUP=(7,NO)
TERMINAL ,NAME=TERMA
TERMINAL ,NAME=TERMB,BACKUP=NO
TERMINAL ,NAME=TERMC
```

Example 2: If you want all the terminals defined by a TYPE macro to be Class 2 terminals with the session recovery priority of 7, specify:

```
TYPE . . .BACKUP=(7,NO)
```

Example 3: If you do not want IMS to recover sessions on a Class 1 terminal at a takeover, specify:

```
TERMINAL . . .,BACKUP=NO
```

Defining MSC Links

If XRF IMS communicates with another IMS subsystem over MSC links, you can change the priority with which IMS establishes new sessions on these links.

The BACKUP keyword on the MSPLINK macro defines MSC physical link recovery and sets the priority for the establishment of new sessions on these physical links at takeover. Consider raising the priority that IMS uses to establish new sessions across these links. The default is BACKUP=4.

The BACKUP keyword on the MSPLINK macro allows you to override the BACKUP option you specified on the MSPLINK macro for the logical link definition. The default is BACKUP=4.

Using IMS.PROCLIB Members

Each subsystem has IMS running continuously. You initially declare which is the active IMS with the HSBID parameter in the EXEC statement. The HSBID parameter, HSBID=1 or HSBID=2, points to the respective IMS APPLID in the COMM statement, the ADDR for a BTAM master terminal, or the NODE for a VTAM master terminal. The HSBID is also used to identify all the message queue data sets associated with each subsystem.

You also need to specify, with the HSBMBR parameter, a suffix that corresponds to a DFSHSBxx member included in IMS.PROCLIB.

The DFSHSBxx member specifies the name of the XRF complex as it is known to the MVS availability manager, to DBRC, and to the IRLM. It also specifies the surveillance mechanism and the takeover conditions for the complex. It is recommended that both the active and alternate subsystems use the same member.

Related Reading: For information on coding the IMS PROCLIB options, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Coding Parameters in DFSHSBxx

When you tailor the IMS system, XRF requires that you specify an IMS control statement in member DFSHSBxx of IMS.PROCLIB. The control statement consists of a number of parameters. The following sections describe the parameters, beginning with the important topic of surveillance for the installation.

You have one copy of DFSHSBxx for the active IMS and one for the alternate IMS. The two copies need not be identical, but the USERVAR and RSENAME parameters must be the same in each. HSMBR=xx on an IMS.PROCLIB procedure (PROC) statement defines the DFSHSBxx member.

Although IMS provides commands that operators can use to dynamically change the surveillance options in DFSHSBxx, other parameters in DFSHSBxx change only when MVS starts a new IMS subsystem.

Related Reading: For more information on coding these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Figure 38 shows the installation step in which you improve the performance of IMS by defining operating parameters. It shows the parameters that you specify in IMS.PROCLIB member DFSHSBxx.

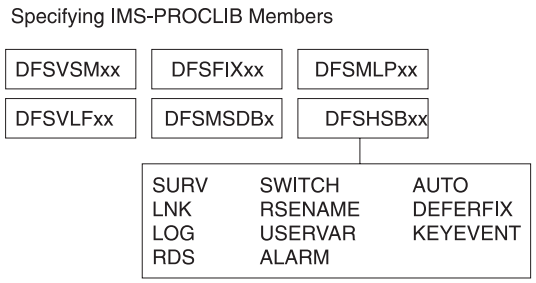


Figure 38. XRF Parameters for Member DFSHSBxx of IMS.PROCLIB

This section first describes the parameters that determine surveillance—the signals that warn the alternate IMS of possible failures in the active IMS—and then describes the other parameters.

RSENAME=

The recoverable service element (RSE) consists of the active and alternate subsystems. Although failures of VTAM, MVS, and the CPC can cause a takeover, these elements of the XRF complex depend on IMS to recognize a failure and initiate a takeover. The RSENAME is a 1- to 8-character ID. It is through this ID that IMS is known to DBRC, IRLM, and the OS/390 availability manager. Signon and authorization of DBRC databases is done by the RSENAME keyword. RSENAME is the ID used to notify the availability manager of the active and alternate subsystems. It is also the name the operator sees in the availability manager messages. RSENAME is required for XRF.

USERVAR=

The USERVAR parameter defines the USERVAR for IMS. It resides in the USERVAR table. The table contains the APPLID of the active IMS subsystem. Users logging on to the XRF complex know only an IMS logon name. End users logging on need only know the logon message to be connected to the active subsystem. When a takeover occurs, IMS places the APPLID of the new active subsystem in the USERVAR table of the new active IMS. Other VTAM domains without automatically managed USERVARS must be notified of the

change either by the operator or via an NCCF CLIST. Notification is unnecessary if these VTAMs are Version 4 level or higher and have the USERVAR management enhancement.

SURV=ALL,LNK,LOG,RDS,NO

XRF IMS provides surveillance to inform the alternate IMS of failures in the active IMS. Without surveillance, some failures in the active IMS occur without any warning to the alternate IMS. The SURV parameter specifies what surveillance mechanisms are to be in effect for the XRF complex.

Options on the SURV parameter are:

Option Result

ALL The ALL parameter indicates that all three surveillance mechanisms (LNK, LOG, and RDS) are to be used.

LNK The LNK parameter indicates that the active IMS is to send signals across an ISC link to the alternate IMS at the frequency specified by the interval value on the LNK parameter. The ISC link is also used to request the SNAPQ checkpoint of the active IMS from the alternate IMS during synchronization.

LOG The LOG parameter indicates that the alternate IMS is to check the OLDS for new records as often as the interval value on the LOG parameter indicates. This check is in addition to the alternate IMS's access to the OLDS to maintain a system status on the alternate IMS which reflects the processing that is occurring on the active IMS.

RDS The RDS parameter indicates that the active IMS is to place a timestamp in the shared restart data set (RDS) as often as the interval value on the RDS parameter indicates. The alternate IMS checks the RDS to ensure that the time stamps are being written.

NO This parameter indicates that no surveillance is active.

Any combination of LNK, LOG, or RDS can be specified but using all three is recommended. You can dynamically change the surveillance and the interval parameters by commands issued from the master terminal.

Each of the three surveillance mechanisms (LNK, LOG, and RDS) is specified with an interval and a timeout value. On the active IMS, the interval value indicates the frequency at which the surveillance mechanism sends a signal or updates a data set. On the alternate IMS, the interval indicates the frequency at which the signals or time stamps are received. If the timeout value is exceeded, the alternate IMS can initiate a takeover. These time values are only approximate; actual values might be slightly longer.

These surveillance parameters determine:

- What surveillance mechanisms operate in the complex
- How often the alternate IMS receives signals from surveillance (LNK, LOG, and RDS)
- Whether the absence of any of the surveillance mechanisms causes a takeover

The alternate IMS uses these three surveillance mechanisms in different ways. The ISC link (LNK) is a low-overhead surveillance mechanism that is best used in conjunction with the LOG or RDS options. If the ISC link fails or the RDS incurs a write error, the active IMS writes log records to indicate the failure.

Failure of the alternate IMS to receive signals on the ISC link or the RDS is possible cause for failure; failure of the alternate IMS to receive new records on the log is not. For example, suppose you specify SURV=LNK,LOG, and the alternate IMS stops receiving signals on the ISC link. The failure of signals over the ISC link indicates a takeover, but as long as the log continues to send records satisfactorily, IMS does not request a takeover. In this case, IMS assumes that the ISC link itself is experiencing difficulty and is not a reliable indicator.

Do not confuse the alternate IMS's reading of the log with LOG as a surveillance mechanism. XRF IMS requires that the alternate IMS read the log; XRF IMS depends on information in the log to update its control blocks. Only when you specify SURV=LOG does the alternate IMS periodically check the log to ensure that the active IMS is sending new records. Failure of the alternate IMS to receive new records does not cause a takeover. IMS uses the LOG option to confirm or veto a decision that the alternate IMS might make based on the lack of signals from the RDS or the ISC link. Do not use the LOG option as the only surveillance mechanism; it is best used to validate that the system is indeed not working if the LNK or RDS fails. Absence of LOG records might simply mean that no work needs to be done during the specified interval.

The active IMS writes to the log and the alternate IMS reads the log at two different rates. If the alternate IMS is not caught up, it reads the log continually. When it is caught up, the interval value on the LOG parameter controls the frequency with which the alternate IMS reads the log. Therefore, this interval can affect the level of I/O activity. One way to reduce the work of the alternate IMS is to increase the interval you specify on the LOG parameter. If you set the interval too low, I/O activity to data sets increases, which can affect the performance of the active IMS.

The RDS is an important surveillance tool. If the ISC link fails or the system is inactive and no log records are written, the RDS continues to be updated. For example, if you specify SURV=LNK,RDS and MVS enters a wait state, the ISC link and the RDS cease sending periodic signals. This warns the alternate IMS of a problem. If you specify no surveillance, the alternate IMS does not receive warning of the wait state. In that case, an operator must notice this condition and request a takeover. Table 32 shows how the alternate IMS uses the RDS, the IMS link, and log records to learn of certain events.

Table 32. Informing the Alternate IMS of Events in the Active IMS

If This Event Occurs	Does a Log Record Tell the Alternate?	Which Surveillance Option Informs the Alternate?
IMS abend	Yes	None
MVS abend	No	LNK, RDS
MVS loop[1]	No	LNK, RDS
MVS wait state[1]	No	LNK, RDS
CPC failure	No	LNK, RDS
VTAM failure that takes TPEND exit	Yes	None
VTAM wait	No	LNK,RDS
IRLM failure	Yes	None

Table 32. Informing the Alternate IMS of Events in the Active IMS (continued)

If This Event Occurs	Does a Log Record Tell the Alternate?	Which Surveillance Option Informs the Alternate?

Note:

1. Surveillance-detectable loops and waits.

Although using the LNK and the LOG as surveillance requires no effort beyond specifying the initial control statements, you might not have an ISC link already in place. In this case, establish the ISC link through the 37x5 Communication Controller that controls your Class 1 terminals.

If you do not specify the SURV parameter, IMS uses all three options: LNK (the ISC link), the RDS, and LOG.

The default frequency of signals from the active IMS for the ISC link is 3 seconds and for the RDS is 1 second. Two factors to consider in setting the timing for LNK and RDS are the speed of communication over the ISC link and the performance overhead on MVS in the alternate IMS.

The default frequency for the alternate IMS to check for new records in the log is 1 second.

SWITCH=

The following list describes the options on the SWITCH parameter and describes how the alternate IMS uses the option in considering a takeover.

Option Alternate IMS considers a takeover when:

IRLM A log record tells it that the IRLM has failed.

LNK Signals on the ISC link fail to appear after a specified time.

LOG New records on the log fail to appear after a specified time.

RDS Time stamps on the RDS link fail to appear after a specified time.

TPEND
A log record tells it that VTAM has taken an IMS TPEND exit and failed.

To establish failure of VTAM as a cause for takeover, specify the TPEND option on the SWITCH parameter. If many of your terminals are BTAM terminals, you might choose to allow VTAM to fail.

If you use IRLMs to manage local locks or to manage the resources the XRF complex shares with other IMS subsystems, you might want failure of the IRLM to be a cause for takeover. Specify the IRLM option on the SWITCH parameter. If you want takeover on IRLM, specify AUTO=YES.

Specify the surveillance mechanisms on the SWITCH parameter that are critical in alerting the alternate IMS of certain failures. You can combine these options and use them more than once. For example, if you specify SWITCH=(LNK,LOG), IMS considers a takeover if the ISC signal fails to appear and new log records fail to appear after a specified time. One of these events without the other does not cause IMS to consider a takeover.

Specify a timeout number on the LNK, LOG, and RDS parameter that tells the alternate IMS how long it should wait for a signal before considering a takeover.

The default timeout is 9 seconds for the ISC link and 3 seconds for the LOG and the RDS. These times are only approximate; actual values might be slightly larger.

The default for the SWITCH parameter is:

```
SWITCH=(LNK,LOG,RDS),(TPEND),(IRLM)
```

The SWITCH parameter tells the alternate IMS to consider a takeover if VTAM fails and if the result is an IMS TPEND exit, or if the IRLM fails and the result is an IMS STATUS exit, or if all three surveillance mechanisms alert the alternate IMS of problems.

When you specify the timeout interval, consider the following exceptional conditions:

- Software or hardware problems might cause the system to stop processing IMS work for 30 to 60 seconds.

  Certain software or hardware problems might cause wait states or loops that last up to a minute in duration. Examples of these kinds of problems are software recovery routines, hardware spin loops, or hardware recovery procedures. If your timeout interval is set at the default of 9 seconds for LNK, 3 seconds for RDS or 1 second for LOG, these conditions cause a takeover. If the problem becomes chronic, you must identify the cause and make the necessary changes. Before you make the correction, you can either:

  – Raise the timeout intervals on the LNK, LOG, and RDS parameter statements so that the condition does not cause a takeover
  – Specify NO on the AUTO parameter to allow the operator to take control at takeover, giving the condition more time to correct itself and possibly avoid a takeover
- A SLIP trap specifies that a restartable wait state be entered when the trap conditions are met. This stops the active IMS long enough for the alternate IMS to take over. The restartable wait state causes a takeover. It seems likely that a takeover is appropriate in this situation. When the trap occurs, a lengthy analysis period and possibly a stand-alone dump follows.
- An operator stops a CPC

  Surveillance should be stopped when the operator wants to stop the CPC. The operator might do this when entering new JES, because of an SDUMP or a disabled console. If your operator stops the active CPC and has not stopped surveillance, the lack of signals from surveillance mechanisms causes the alternate IMS to request a takeover. If you do not want a takeover to occur, discontinue surveillance or specify AUTO=NO to give control to an operator before a takeover proceeds.

Because takeover can happen when a set of time-dependent parameters are satisfied, setting up surveillance intervals is an important administrative task. Also, those intervals might not be suitable for all workloads run by the active IMS subsystem.

Factors that affect surveillance intervals are:

- CPC power or type (for example, UP or MP)
- Operational procedures and response objectives
- Recovery/delay situations where a timeout is exceeded
- For critical work, the relative tolerance to unplanned outage

ALARM=NO|YES
This parameter requests that a service processor notify the operators when a takeover begins. The default is NO.

When an automatic takeover of the active IMS by the alternate IMS occurs, the service processor alarm on the alternate IMS sounds if YES has been specified. The alarm also sounds when operator intervention is needed to manually switch the system.

AUTO=YES|NO
AUTO establishes whether the takeover proceeds automatically when IMS requests it or whether the operator intervenes. If you specify AUTO=YES, a takeover occurs automatically (based upon a failure in the host as detected by the surveillance mechanism). If you specify AUTO=NO, the operator must manually force the system to switch. Message DFS3869 appears on the master console of the alternate IMS when the alternate IMS detects, via the surveillance mechanism, that a takeover is indicated. When AUTO=NO is specified, the operator can delay the takeover while trying to resolve the problem on the active IMS. Normal tracking continues on the alternate IMS while the problem on the active IMS is being fixed. The takeover message ceases to be sent to the active IMS after the problem is corrected or after the operator initiates the switch.

Operator intervention undoubtedly extends the time before NCP switches sessions on XRF terminals and IMS reestablishes sessions on terminals. Operator intervention also causes disruption of the workload processing. Your installation, though, might want the operator to perform certain tasks before the takeover proceeds. The tasks might be:

- Ensuring that the takeover decision is valid
- Manually switching terminals or DASD
- Quiescing of jobs on the alternate IMS

If it is important to control the quiescing of the work in the alternate IMS, request that the takeover not proceed without operator intervention. This gives your operator time between the takeover request and the takeover to cancel the jobs.

If you do not want the operator to control the takeover, specify AUTO=YES. The default is AUTO=NO.

KEYEVENT=MSG|NONE
KEYEVENT determines whether your IMS operators receive all of the IMS messages for XRF. Although the takeover messages must be displayed, you can choose to have certain messages suppressed. When you specify KEYEVENT=MSG, all messages are displayed. When you specify KEYEVENT=NONE, optional messages DFS3882 through DFS3888 are suppressed. The default is NONE.

DEFERFIX=xx
This parameter indicates the DFSFIXxx PROCLIB member (see “DFSFIXxx” on page 225) that should be processed if page fixing of non-Fast Path resources on the alternate IMS is to be deferred until takeover. IMS ignores Fast Path page-fixing options at this time. For the alternate IMS, only those DL/I blocks and routines that can be page fixed with the existing DFSFIXxx IMS.PROCLIB member are page fixed.

DFSFIXxx

The DFSFIXxx PROCLIB member specifies the page-fixing values for an IMS system. In an XRF complex, two members can be used. One member specifies what is to be page fixed at initialization time for both the active and alternate IMS subsystems. This member is pointed to by the FIX=xx parameter of the startup procedure. The second member is pointed to by the DEFERFIX=xx parameter in DFSHSBxx member of IMS.PROCLIB. For the active IMS, page fixing requested in this member occurs at initialization. For the alternate IMS, page fixing requested in this member occurs at takeover.

Page fixing done on the alternate IMS during initialization speeds the takeover but takes additional real storage on the alternate IMS throughout the tracking phase.

DFSVSMxx

The DFSVSMxx PROCLIB member contains the log data set definition information, specifies the allocation of OLDS and WADS, the number of buffers to be used for the OLDS, and the mode of operation of the OLDS (single or dual). Dual OLDSs are recommended, because when loss of the OLDS occurs and no backup is available, the XRF complex fails. Because the alternate IMS allocates the log data sets defined to it and any data sets used by the active IMS, the mode of operation and buffer definitions must be the same for both subsystems. Both active and alternate IMS subsystems must use the same member.

Placement of IMS Data Sets in the XRF Configuration

Although IMS running with an alternate IMS in another CPC is managed operationally as a single system, you do need to plan in detail for duplication of IMS system data sets.

The three main considerations for placing your data sets are:

- Availability of data sets during tracking and takeover

  An XRF complex consists of two systems that must sometimes access the same data sets or identical copies of the same data sets. Therefore, IMS requires that you place some data sets on DASD shared by the two systems. It recommends that you place other data sets on shared DASD; you can, however, switch some data sets through a switching device or maintain separate copies of them.
- Prevention of single points of failure

  IMS requires that you maintain (and constantly synchronize) separate copies of some data sets for the two systems. It recommends that you maintain separate copies of other data sets.
- Accessibility of data sets to one IMS system

  IMS recommends that you keep the data sets that are unique to one system on local (that is, nonshared) DASD.

For the best performance at takeover, keep the shared and nonshared I/Os separate from each other. For example, place the IMS database data sets on different volumes, controllers, and channels than the MVS data sets.

Placement of the system logs is important. A critical step in the takeover process is the isolation of the databases from the failing active IMS. When the active IMS and the alternate IMS run on different CPCs, the alternate IMS can stop the failing active IMS from accessing the OLDS and WADS by reserving the DASD devices on which they reside. Therefore, do not place other data sets on the DASD that contain the OLDS and the WADS.

You should dynamically allocate all IMS databases and area data sets. If they do not already exist, you must generate the DFSMDA members for all IMS databases and areas. All IMS full function database names and DEDB area names must be unique.

Related Reading: For more information on generating the DFSMDA members for all IMS databases and areas, see "Chapter 3. Defining Your System" on page 43.

Some of the requirements and recommendations in this section address the protection of your data sets. Always protect your data sets to the extent your resources allow. Figure 39 summarizes the required and recommended placement of these data sets.

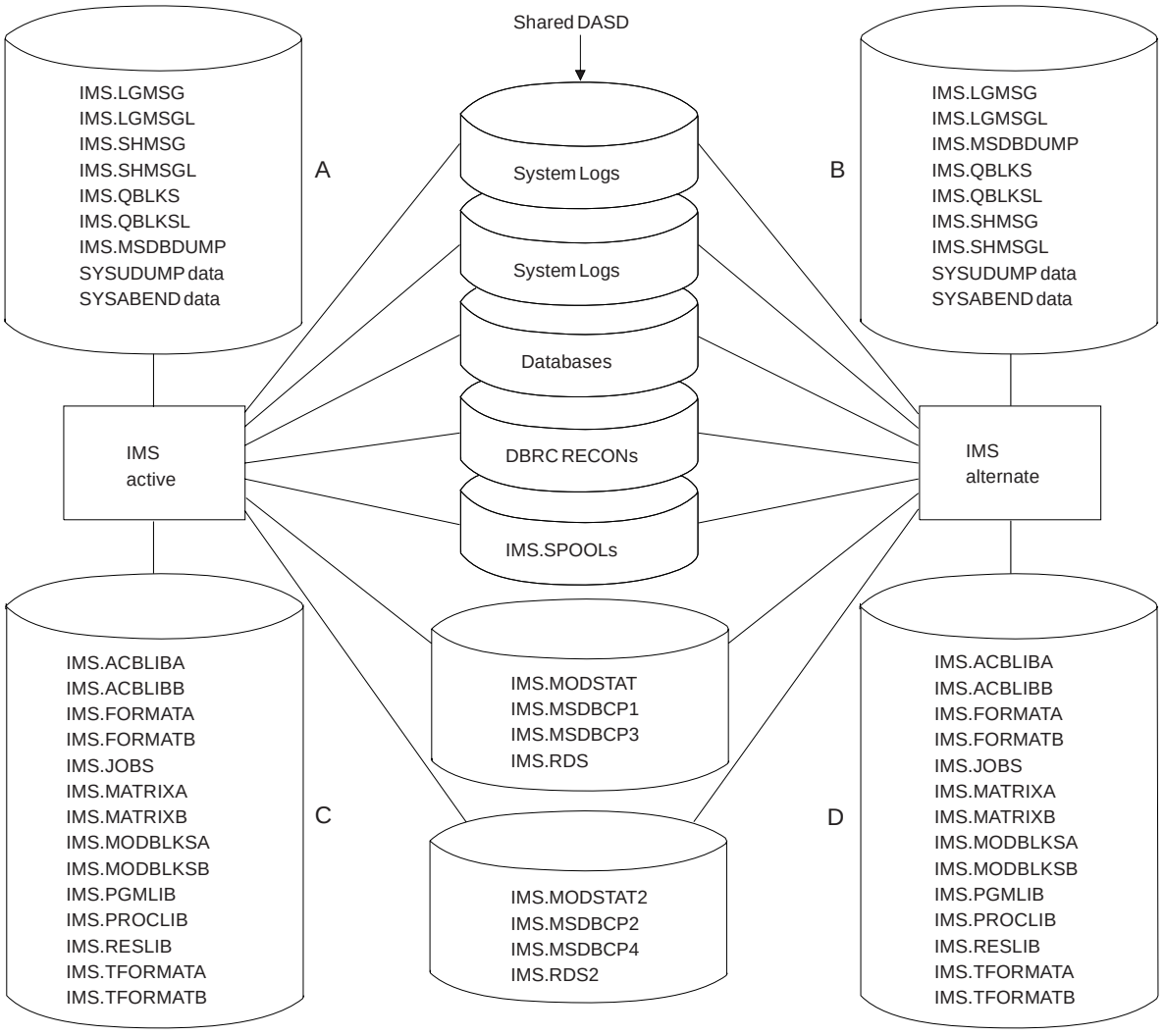


Figure 39. Recommended Data Set Placement

Additional Data Sets Required for XRF

The additional data sets listed in Table 33 on page 227 are present in control region JCL in an XRF complex.

Table 33. Additional Data Sets Required by XRF

Data Set	Usage
IMS.RDS2	Separate restart data set
IMS.LGMSGL	Local message queue
IMS.SHMSGL	Local message queue
IMS.QBLKSL	Local message queue
IMS.MODSTAT2	Separate MODSTAT data set
IMS.MSDBCP3	Alternative MSDB checkpoint
IMS.MSDBCP4	Alternative MSDB checkpoint

Data Sets That Must Be Shared in the XRF Complex

In addition to critical data sets that make up databases, several system data sets need to have a single copy and be shared by both the active and alternate subsystems. These have the following ddnames:

- RECON1, RECON2, RECON3
- DFSOLPnn
- DFSOLSnn
- DFSWADSn
- IMS Spool data sets
- IMS.MODSTAT
- IMS.MODSTAT2
- IMS.RDS
- IMS.RDS2
- IMS.MSDBCP1
- IMS.MSDBCP2
- IMS.MSDBCP3
- IMS.MSDBCP4

For added efficiency and ease of operation, it is recommended (though not required) that the DL/I databases be on DASD shared by the two CPCs. You might, however, choose to switch these data sets through a switching unit, such as an IBM 3814, or a channel-switch on an IBM 3880 Storage Control.

It is also recommended that you have two copies of the OLDS shared by the two CPCs. If you have one copy of the OLDS and a permanent I/O error or power failure causes physical loss of the one copy, tracking cannot continue. In this case, a failure of the new active IMS occurs.

Figure 39 on page 226 identifies the data sets that you should place on shared DASD. In the figure, they are on the DASD located between the active IMS and the alternate IMS. Notice that the system logs, the RDS, MODSTAT, and MSDB data sets are not only on shared DASD, but separate copies also reside on shared DASD. The active IMS maintains these copies.

Data Sets Not Shared in the XRF Complex

For best performance, place data sets that contain information unique to one system on local, nonshared DASD, and define these data sets in separate catalogs. The data sets you maintain separately contain information unique to one system, such as the data a system needs when it attempts to recover from a failure.

Figure 39 on page 226 shows these data sets on DASD labeled A and B. They could also be on DASD that is shared by the two systems.

Maintain certain data sets as distinct library copies that are not to be shared. This is to reduce the points of failure. These system data sets are:

- IMS.ACBLIBA(B)
- IMS.FORMATA(B)
- IMS.TFORMATA(B)
- IMS.MATRIXA(B)
- IMS.MODBLKSA(B)
- IMS.JOBS
- IMS.PROCLIB
- IMS.RESLIB

Data Sets That Must Be Duplicated

Some data sets must also be separate data sets. Those system data sets are:

- IMS.MSDBDUMP
- IMS.LGMSG
- IMS.SHMSG
- IMS.QBLKS
- IMS.LGMSGL
- IMS.SHMSGL
- IMS.QBLKSL
- IMS Spool data sets
- SYSUDUMP and SYSABEND data sets

Making Online Changes in an XRF Complex

You can still use the Online Change utility in the XRF complex. When you use it to maintain separate but duplicate copies of IMS data sets, make the identical changes to both copies of a data set. Then issue the `/MODIFY PREPARE` and `/MODIFY COMMIT` commands at the active IMS.

Part 2. System Management

Chapter 6. Testing Your System

As a system administrator, you need to be involved in two phases of testing. One phase occurs when application programs are verified prior to a cut over to production mode. The second phase involves evaluating changes or corrections to existing application programs to ensure that application function has not regressed and to validate the new or changed function.

When you have multiple application programs, you must ensure that modifications to one application program do not impact the service provided to any end users.

The various testing phases, with the corresponding administration tasks and related development activity, are listed in Table 34.

In this Chapter:

The Need for a Test System

Your primary concern at the final stage of implementation is to offer satisfactory service to the end users. Does the IMS system perform as expected?

A secondary concern, when changes occur in one or more online application programs, is to preserve the integrity and service for all end users. Will changes seriously impact the production environment?

The answer to both of these concerns is to have a test system in place. The development of such a system requires the participation of representatives from several areas. Your installation might have a separate test organization. In this case, development personnel do not test the working code but might have to demonstrate it before handing over the application programs for independent testing.

The end user might be represented by a user-liaison group that knows the business needs. The liaison group also might evaluate the adequacy and accuracy of application programs.

Your solution to the need for a testing system can take several forms:

- A separate IMS test environment that is operational during major development activity
- A separate IMS test system used for ongoing verification of maintenance and application design changes
- A production system that allows controlled changes to be tested online, possibly using regions that are initialized for test purposes at a time that does not impact production processing

Table 34. Administration Tasks Related to Testing Phases

Test Phase	Administration Task	Related Development Activity
Unit test	Identify bottlenecks	Test development code

Table 34. Administration Tasks Related to Testing Phases (continued)

Test Phase	Administration Task	Related Development Activity
Function test	Plan operations procedures	Test explicit functions
Integration test	Prepare test system definition	Build the system
	Plan test database	
Component test	Verify network operation	Validate major portions of application program logic
System test	Check out operations and recovery procedures	Build fully executable system
	Coordinate use of test tools and monitoring	Validate operational procedures
	Ensure network readiness	Test coexistence
Performance testing	Plan for simulation	Validate claims and set benchmarks
Stress testing	Plan for peak loads and response criteria	Test for high volume of traffic and processing
Acceptance test	Finalize operations procedures	User liaison checkout on behalf of end users
Maintenance testing	Control libraries and online definition	Application and IMS service checkout
Design change testing	Plan response across administration tasks	Control verification of changes
Regression testing	Plan monitoring	Verify that old function is not damaged

Part of your administration role is to ensure that a suitable online IMS test system is created and that the test procedures and any special database requirements are well documented. This is important if the test system is also to function as the maintenance system. However, your role can be more passive, in which you only participate in testing to verify operations procedures and to verify that all system definitions and preparation for production mode are in place.

After the application package has entered the production phase, the role of the developer or tester is usually taken over by program maintenance personnel. Administration needs to assess the impact of maintenance to the existing online IMS design. This task is described in more detail in “Chapter 9. Modifying Your System Design” on page 295.

Setting Up a Test System

The major considerations for establishing a separate test system are:

- The installation’s policy for protecting the integrity of the database
- The presence of criteria for accepting an application as properly tested
- The need for a stable production environment
- The degree of support available to solve day-to-day problems
- The complexity of the applications
- The degree to which the current programs can perform correctly, or function with interim problem bypasses.

Some of the factors involved in the decision to use a test system as an ongoing function are:

- Will there be staged implementation of new application programs or add-on function?
- Can online service interruptions caused by the need to fix application problems encountered during testing be tolerated and, if so, to what degree?
- What procedures should be followed for accepting system changes or corrections?
- How is maintenance to the IMS system itself to be handled—will the test system be used?
- Will corrections to the application programs and IMS problem fixes be incorporated on demand or batched at agreed-upon intervals?

An administration task is to define the procedures for applying changes to the online IMS system. Part of this task is to have an acceptance procedure that is followed by all responsible participants.

Setting Up a Test Database

A test database must contain enough data to adequately test the system. The data should exercise a major portion of an application program's logic to prevent regression of existing function and yet economize on the amount of data.

Related Reading: For more information on creating test databases, see *IMS/ESA Administration Guide: Database Manager*.

Usually, a test transaction stream executes against a known database status, and the results are compared to predicted results. The content of the transactions and implications of the database status needs to be analyzed for accuracy by a user-liaison, group as well as by groups responsible for the database design. As additional test cases are added, they need to be validated and inspected for redundancy. The trade-off is between adding to an existing transaction or defining an additional transaction that might require extra database content. The test case stream must be adequately documented, so that you know what it does and what data it needs.

Testing Operational Procedures

You can use the test system to ensure the accuracy and usability of your operational procedures. The following activities are suitable during the system test phase:

- You can test preliminary versions of the MTO procedures, the run book, and the incident report forms. The MTO can perform system startup, connect a subset of the terminals and nodes, and practice the restart actions.
- You can hold an informal audit of command use and discuss any misunderstandings to help you refine the content of the operator instructions.
- During online execution, you can arrange other events that require MTO intervention to take place, such as taking image copies, invoking the IMS Monitor and traces, and responding to application program abnormal termination.
- You should also test the recovery procedures. Have the operations staff carry out a database recovery and verify the correct use of system logging control and recovery utilities. You can build a test RECON data set for DBRC and follow through the GENJCL step to recover to a given checkpoint position.

Test System Tasks

If the test system is used for checkout of changes to the application package, any significant operational changes should also be exercised, possibly with MTO observers.

Monitoring in IMS Test Environments

Considerations and tools for your monitoring strategy are described in “Chapter 7. Monitoring Your System” on page 247. However, your objectives for monitoring during a test phase are slightly different from those for production systems. Three activities you should plan for are :

- Detecting and correcting potential performance problems before entry into production mode.

  Using realistic data, you can expose patterns of processing for the application programs, especially the DL/I call occurrences. Compare the Monitor results against the expected transaction profile. Such items as excessive I/O events or large I/O wait times can reveal a performance problem at an early stage.
- Testing the monitoring part of your operations procedures.

  Become familiar with the way output is produced and the format and content of reports. Start to develop work sheets to summarize Monitor findings rather than write unstructured comments on the report output. Refer to the detailed report descriptions given in “Chapter 7. Monitoring Your System” on page 247.
- Using your monitoring tools for base profiles of new program processing.

  Discuss your findings with development personnel and with performance specialists. Try to obtain early warning of performance problems. If you are integrating a new application into an existing online IMS system, try to assess beforehand the impact of the added application workload.

During the testing stage you can also perform stand-alone or calibration runs to establish base profiles for the critical transactions. The Call Summary report from the IMS Monitor is useful for this purpose. You can also compare the results for the tested transactions to similar transactions already in the production system.

Monitoring in a DB/DC Environment

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		

In a DB/DC environment, you can obtain information about the precise sequence of calls issued by application programs using the report capabilities of IMSASAP II. This tool requires using the online IMS Monitor to produce the monitor trace records. Its use as a monitoring tool and installation prerequisites are described in “Chapter 7. Monitoring Your System” on page 247.

Ensuring Network Readiness

When you prepare an application package for production mode, you must be aware of the status of all terminals or connected devices planned for availability with the online IMS system. Develop a detailed implementation plan for each device and control unit. Some of the items in this document might be:

- Exact type of device, model, and operational characteristics
- The configuration for a set of components
- Physical location and person responsible for the device installation
- The names by which the online IMS system knows the device: LTERM, line number, unit address, and node names

- The correct VTAM MODEENT macro's PSERVIC parameters for: LUNAME, LUTYPE, TS Profile, 32xx model, 3270 screen size, and NTO device type
- The logon, user, and MSC descriptors for terminals and users defined with ETO
- The appropriate exit routines
- If an output device, the source of paper supplies and arrangements for distribution of output
- If already in operation outside of IMS, the restrictions on the use of the device for test purposes
- System generation requirements for the device, and the planned timing of the generation
- If the device is programmable, what programs are necessary, locally and in the host, for IMS execution

Use the terminal profiles built up as part of the preparation for system and network definition. You need this information because of the potentially large number of devices that can operate in the online IMS system. Also, you need to converse in terms of the hardware with installation personnel and data communications specialists whose responsibilities usually lie beyond IMS device support.

Contact the person responsible for the system generation regarding the status of statically defined VTAM terminals. The generation events are usually scheduled as part of a master plan for implementing the application package.

Network Testing

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

As part of your preparation for production mode, you should arrange for online testing of each piece of the network during the system test. Try to observe the online testing for the different terminal types that are to be active in the IMS system. Include observations of those transactions considered critical. The results of this exercise are useful when you write operations procedures for remote terminals and instructions for the master terminal operator. Emphasizing the physical actions and sequence of events helps clarify these procedures and make them more complete for the end user.

If you do not have access to an early draft of the actual procedure for the terminal operation, it is best to use a prepared script for a session. The text should include a scenario depicting a test sequence of commands as an online session:

1. Prepare to transmit the SNA terminal subsystem program, then transmit from the host (if applicable).
2. Start the ACF/VTAM or host subsystem application program, and enter the appropriate commands to start IMS and establish communication with VTAM.
3. Start the SNA terminal subsystem program at the processing unit location (if applicable) and make the intelligent terminal ready for communication with IMS.
4. Exchange messages with the host, submit an IMS transaction from the terminal, and validate the result.

Testing in a DBCTL Environment

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

If you are testing a DBCTL environment with a CCTL, be aware that the CCTL, not IMS, controls the network, terminals, and transactions.

Testing in a DCCTL Environment

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
			X

You can test any DCCTL function that does not require access to a database. For example, you can schedule transactions or perform component testing if no database calls are made. If your test system already has a GSAM or an external subsystem (for example, DB2) installed, you can verify connections and applications requiring those systems.

IMS Testing Aids

This section provides information on:

- "Simulating Online Execution with Batch Terminal Simulator"
- "Online Testing of MFS Formats" on page 243
- "Using Online Change for Testing" on page 244
- "Program Testing Using SYSIN/SYSOUT" on page 245
- "Network Testing Using Teleprocessing Network Simulator" on page 245

Related Reading: For more information on suggested procedures and utilities used during testing of application program code, see *IMS/ESA Application Programming: Design Guide*.

Simulating Online Execution with Batch Terminal Simulator

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Often an online application program is developed to the testing stage, but a suitable online IMS system is not available. At this stage you can use the Batch Terminal Simulator licensed program, program number 5668-948. The Batch Terminal Simulator (BTS) program can simulate the operation of message processing regions before the online IMS system is operational as a test system. This program runs as a batch IMS system using one or more applications. BTS input is transaction data, and the application programs are invoked. Database calls are executed against test databases; the DL/I calls for data communications are simulated.

With BTS, you can do the following:

- Print terminal input, output, and an optional trace of related database activity. You can request a summary of DL/I calls, by type, against each PCB.
- Simulate conversational transactions and program-to-program switches.
- Simulate the message queuing and application program scheduling functions of the online IMS system. transaction input data to the input message structure for

the application program and, similarly, map the output messages to a printed layout. This is particularly useful for IMS 3270 input and output formats.
- Use the debugging and trace facilities (during the test phase) for both batch and telecommunication applications.

You do not need to modify the IMS control programs, control blocks, libraries, or the application programs.

Related Reading: For more information on BTS, see *Batch Terminal Simulator: General Information Manual*.

Online Testing of MFS Formats

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

If you need to test an application program change online in either a test or a production system and the test transaction input uses an MFS-supported screen format, you can use the MFSTEST mode.

IMS allows individual MFS-supported terminals to enter MFSTEST mode. In this mode, you can use temporary message formats from an alternative MFS format library, rather than changing the message format blocks in the production format library. A format block name can be identical to the name of a message format already in the production library.

An MFS-supported terminal is placed in test mode by entering a `/TEST` command with the MFS parameter. Next, the remote terminal operator enters a `/FORMAT` command for transaction processing, causing message formats to be selected from a library of test formats (IMS.TFORMAT). If these message formats are not found in this library, they will be selected from the active message format library (IMS.FORMATA/B). By entering an `/END` command, the terminal is removed from test mode.

When testing MFS formats, you can use the `/TRACE SET ON TRAP` command to trap and analyze MFS errors. This command helps you analyze errors that result from incorrect manipulation of the MFS control blocks.

System Definition Requirements for MFSTEST Mode

To use MFSTEST formats, specify the appropriate parameter on either the IMSGEN or COMM macro.

Related Reading: For more information on these macros, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Online Execution Requirements for MFSTEST Mode

When you want to perform testing of MFS formats during execution of the online IMS production system, the IMS.TFORMAT library must be defined and must contain the appropriate test blocks. These can be named the same as existing production formats. The IMSTFMTA/B DD statements must be present in the control region JCL, with the IMS.TFORMAT library as the first data set, followed by the corresponding IMS.FORMATA/B library concatenated to it.

The CIOP is used to hold all format blocks for terminals operating in MFSTEST mode. Storage space in CIOP is dynamically allocated according to your system demands.

Executing in MFSTEST mode does result in a performance impact, because both transmission and message formatting are being stored in the same buffer.

Using Online Change for Testing

Using the online change capabilities, you can plan to perform selective testing during execution of an IMS online test system. Your testing can be accomplished without generating a modified test system merely to accommodate some temporary changes.

You can perform testing that supports:

- The addition of new applications
  - New transactions, programs, and MFS formats
  - Additional databases
- Modifications to existing applications
  - Modified database definitions
  - Replaced PSBs and programs
  - Altered MFS formats
  - Necessary deletions for the above modifications

You cannot use this method to test the use of terminals or devices not already defined to the IMS system.

Another factor might be the security arrangements. Planning for the online change must include an update to the Security Maintenance utility input so that appropriate authorization exists to use the transactions or existing terminals for the duration of the test.

You might plan for a group of application changes to make up a test package to be executed during IMS online operation in a test system, or even during a production cycle. Online change is best used to migrate pretested changes without the need for a full restart interruption. Your planning should include the following considerations:

- Online data resources must be adequately protected. Carefully assess the risk that an application program being tested could damage production data.
- Added databases must be included in the JCL for the IMS control region; a solution for MVS systems is to use dynamic allocation.
- Although changes to database structures can be reflected in an ACBGEN, this might not always be practical if the changes require a reorganization.
- You must coordinate the update of IMS.PGMLIB with the status of the online processing. A replaced program might cause problems, unless the end user is aware of the activity. The library update might have to be carried out in the interval just before the `/MODIFY COMMIT` command.
- You must assess the potential performance impact to tuned systems, although a monitored test execution can provide valuable information for performance planning.

The entry of the `/MODIFY PREPARE` command begins the cut-over operation. When activity has stabilized for the resources affected by the content of the online

change, the /MODIFY COMMIT command completes the cut-over. Testing can now begin. When you complete the planned testing, repeat the /MODIFY PREPARE, /MODIFY COMMIT command sequence to reinstate the old unmodified system data sets that were inactive during the testing. It is the responsibility of those performing the test to nullify the effects of the testing and back out any inappropriate changes made to the databases.

Program Testing Using SYSIN/SYSOUT

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

One way of testing a message processing program is to provide an input data stream containing messages. Specify a line group as a READER, then assign the input SYSIN to a local card reader. Messages are passed to the application program by GU calls to the message queue. No editing or logging of position occurs while messages are being processed. The accuracy of the end-of-message or segmentation is determined by the input stream.

Similarly, for output, you can assign a line group as a printer, punch, tape, or DASD device using the UNITYPE keyword for the LINEGRP macro. You can then assign the output LTERM for the appropriate device characteristics, although the actual device does not have to be allocated.

The LINEGRP macro parameters produce a DD statement for the control region, and the appropriate buffer size is defined for the output records. For printer output, a translation to the 48-character set, lowercase to uppercase, occurs; all other codes are converted to periods.

Network Testing Using Teleprocessing Network Simulator

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Teleprocessing Network Simulator (TPNS), program number 5662-262, is a terminal and network simulation tool used for determining system performance and response times, evaluating teleprocessing network design, functional testing, and automating regression test procedures.

Chapter 7. Monitoring Your System

Monitoring is the collection and interpretation of IMS data. Monitoring should be an ongoing task because:

- Monitoring helps you establish base profiles, workload statistics, and data for capacity planning and prediction.
- Monitoring gives early warning and comparative data to help you prevent performance problems.
- Monitoring validates tuning you have done in response to a performance problem and ascertains the effectiveness of that tuning.

An historical base and conclusions from continuous monitoring provide a good start to answering end-user complaints and an initial direction for tuning projects.

In this Chapter:

Establishing Monitoring Procedures

Several types of monitoring strategies are available. You can:

- Summarize actual workload for the entire online execution. This can include both continuous and periodic tracking. You can track total workload or selected representative transactions.
- Take sample snapshots at peak loads and under normal conditions. It is always useful to monitor the peak periods for two reasons:
  - Bottlenecks and response time problems are more pronounced at peak volumes.
  - The current peak load is a good indicator of what the future average will be like.
- Monitor critical transactions or programs that have documented performance criteria.
- Use the MVS Workload Manager to help manage workload distribution, balance workloads, and distribute resources.

Plan your monitoring procedures in advance. A procedure should explain the tools to be used, the analysis techniques to be used, the operational extent of those activities, and how often they are to be performed.

Regardless of which strategy you use, you need to:

- Develop performance criteria
- Develop a master plan for monitoring, data gathering, and analysis

The sections that follow summarize the performance administration activities for these two tasks.

Establishing Performance Objectives

Inherent in the design of your online IMS system are your performance objectives. Establishing performance objectives is a major task requiring data gathering for the IMS workload as a whole. After defining the workload and estimating the resources required, you must reconcile the desired response with the response you consider to be attainable. Monitor the performance of the system to determine if these objectives are being met.Base performance objectives on:

- Desired, acceptable, and maximum response time
- Average and maximum resource demands (or workload) per transaction
- Predicted and actual transaction volumes

Establishing your performance objectives is an iterative process involving the following activities:

1. Defining user-oriented performance objectives and priorities

   These derive from the way an end user perceives the service provided by the system. For IMS these objectives state expectations of response time as seen by the end user at the terminal.

   When response time objectives are established, they should not only reflect the time-in-system (the elapsed time from entry of a last input message segment to the first response segment), but also the expected amount of IMS and application program processing. You should consider whether to define your criteria in terms of the average, the 90th percentile, or even worst-case response time. Your choice depends on your installation's audit controls and the nature of the specific transactions.
2. Determining how performance against these objectives is measured and reported to users

   This includes identification of systematic differences between the measured data and what the user sees. You should investigate the differences between internal (as seen by IMS) and external (as seen by the end user) measures of response time. To do this, you can use the following tools:

   - The Transaction Response report produced by the IMS Statistics Analysis utility, which gives the following data on internal response time by transaction type:
     - Longest and shortest response
     - 25th, 50th, 75th, and 90th percentiles of the response time distribution
   - Installation-written programs, which can analyze the output of the IMS Log Analysis utility. Other programs can also provide the same type of information, and are usually tailored to satisfy the installation's requirements.
3. Understanding and documenting the current workload

   This requires breaking down the total work into categories and, for each category, developing a workload profile (generally estimates) that include:

   - Definition of the transaction category (for example, transaction type or group of transactions). The two characteristics of the category are:
     - The IMS workload, which is generally documented by transaction profile. In a well-designed IMS online system, most transactions perform a single function and have an identifiable workload profile. Later in the process, transaction types with common profiles can be amalgamated for convenience.
     - The transaction volume. In situations where the workload to be documented is already operational, a summary of transaction volumes can

be obtained from the IMS Statistical Analysis utility. (The Application Accounting Report gives transaction counts within program name.) In other cases, volumes are estimated.

- Relative priority of category, including periods during which priority changes.
- Resource requirements of the work:
  - Physical resources managed by the operating system (real storage, DASD I/O, terminal I/O)
  - Logical resources managed by the subsystem, such as control blocks, latches, buffers, and number of regions

To obtain a base profile of transaction resource demands, you can start IMS on a dedicated machine and execute a few transactions to accomplish initialization and buffer pool usage. Then start the IMS Monitor and measure a sample of transaction execution.

You can use the base transaction profile to examine the transaction workload to see if it can be reduced. Such design changes result in the greatest impact, occurring before system-wide contention. You can also compare the base profile to the transaction profile in the production environment.

4. Translating the resource requirements and volume information obtained into system-oriented objectives for each work category

   This includes statements about the transaction rates to be supported (including any peak periods) and the internal response time profiles to be achieved.

5. Confirming that the system-oriented objectives are reasonable

   After initializing the system and monitoring its operation, you need to find out if the objectives are reasonable (given the hardware available), based upon the measurements of the workload. If the measurements differ greatly from the estimates used, you must revise the workload documentation and system-oriented objectives accordingly, or tune the system.

Establishing performance objectives is also a necessary prerequisite to using the MVS Workload Manager. Much of the information gathered when you establish performance objectives can be used as input when planning for workload management.

Planning for Workload Management

MVS Version 5 provides a workload management function to help you manage workload distribution, balance workloads, and distribute resources to competing workloads. MVS provides this support automatically after you specify how you want your workloads processed, using the panel-driven application that the MVS Workload Manager provides.

Using Workload Management with IMS

With the MVS Workload Manager, you define to MVS performance goals, assign a business importance for incoming IMS transactions, and then let the system decide how the resources that are controlled by MVS should be allocated. A performance goal can be the average response time desired for a transaction, or it can be that a certain percentage of the transactions complete within the response time period. The business importance is a priority level representing how critical a type of transaction is to your installation (with 1 as the highest priority level).

After the transaction is scheduled by IMS, the MVS Workload Manager uses the performance goals you define for each transaction and decides how much resource, such as CPU and storage, should be allocated to meet your goals. When

contention for system resources occurs, the MVS Workload Manager uses the business importance assigned to the transaction to help decide which transactions get priority for the resources that are under the control of the MVS WLM. The performance goals and business importance are defined in a service definition.

Defining WLM Service Classes for IMS Transactions

It is strongly recommended that you specify multiple WLM service classes, differentiating by business importance and goals. Then classify your IMS transactions in the WLM policy by using the IMS transaction class as the work qualifier for the WLM service that best fits the response time goals of each transaction. By assigning different WLM service classifications to different IMS transactions, you insure that the WLM manages the MVS resources in a manner that gives all IMS transactions the best possible chance of obtaining their response time goals

If you assign a single WLM service class to all of your IMS transactions, the WLM treats all IMS address spaces, including the Control Region, equally. This means that the WLM assigns the same dispatch priority to all of these address spaces. If some of your transactions do not make their response time goals, the WLM does not know which address space to give the higher priority to (it should be the IMS Control Region), so the WLM does not change the priority for any of them but continues to give all of the IMS address spaces the same dispatch priority.

Establishing the Service Definition

A *service definition* contains all of the information necessary to perform workload management processing. Workload management provides an online panel-driven application for establishing a service definition. Much of the information you need to establish a service definition is contained in the performance objectives described in “Establishing Performance Objectives” on page 248. An important piece of information contained in the service definition is the classification rule.

A *classification rule* consists of a work qualifier and a service class. One set of classification rules exists for each service definition. However, you can have multiple service classes for each service definition. Using the workload management-supplied function, you establish the classification rule and define the service class by specifying one or more work qualifiers. The *work qualifier* associates incoming transactions with a particular service class, which represents a group of transactions with similar performance criteria (performance goals and business importance, for example). Work qualifiers can be the subsystem type, the IMS transaction name, the IMS transaction class, the inputting LTERM name, or the user ID. You can use any one of these values, or a combination of them, to assign the service class to a transaction.

A *service class* has performance criteria defined for it. After a transaction is assigned a service class, the MVS Workload Manager processes it according to the performance criteria defined for that service class.

Example of a Classification Rule

In the following example, the classification rule uses the transaction name as the work qualifier. It assigns all incoming transactions with the name of DEPOSIT to the service class IMSHIGH.

```
--- Work Qualifier ---      --- Service Class ---

  Subsystem Type = IMS          IMSHIGH
  Tran name = DEPOSIT             Performance goal=  response time
                                                     of less than second

                                  Business importance = 1
```

Figure 40. Example of a Classification Rule

The performance goal for the IMSHIGH service class is a response time of less than 1 second. The business importance of 1 gives the DEPOSIT transactions priority, after they are scheduled by IMS, if contention for system resources occurs.

Migrating to Workload Management

Workload management offers two operating modes: compatibility and goal. Compatibility mode is your present method of performance management, and goal mode is the workload management method. To facilitate migration, you can establish your service definition while running in compatibility mode. Then, after you are comfortable with your service definition and have completed the steps for migrating to workload management, you can switch to goal mode.

Related Reading: For more information on setting up service definitions and using workload management, see *MVS/ESA SP Version 5 Planning: Workload Management*.

Interpreting MVS WLM Change State PB Service Codes

The WLM Change State Performance Block (PB) service is used to show the current state of the transaction. The PB service codes are interpreted as follows for IMS:

State	Description
Active	The transaction is executing an application program.
Free	This is not reported by WLM.
Idle	The transaction is waiting for work.
Waiting - I/O	IMS waiting on I/O (IMS initialed I/O).
Waiting - Lock	IMS waiting on a lock request.

Deciding on Monitoring Activities and Techniques

When you develop a master plan for monitoring and analyzing performance, you should establish:

- A master schedule of monitoring activity

  Coordinate monitoring with operations procedures to allow feedback of online events to be incorporated into instructions for daily or detailed data gathering.
- Which tools are to be used for monitoring

  The tools used for data gathering should provide for dynamic monitoring, a daily collection of statistics, and more detailed monitoring.
- The kinds of analysis to be performed

  Document what data is to be extracted from the monitoring output, identifying the source and usage of the data. Although the formatted reports provided by the

monitoring tools help organize the volume of data, you should probably design worksheets to assist in data extraction and reduction.

- A list of the personnel that are to be included in any review of the findings

  The results and conclusions from analyzing monitor data should be made known to the user liaison group and to system performance specialists.

- A strategy for implementing changes to the online IMS system design resulting from tuning recommendations

  Coordinate this change implementation strategy with your installation's standards for testing and standards for frequency of production environment changes. Change management is described in "Chapter 9. Modifying Your System Design" on page 295.

Plan for three broad levels of monitoring activity:

- Dynamic
- Daily
- Detailed

Dynamic Monitoring

Observe the system's operation continuously to discover any serious short-term deviation from performance objectives.

The output from `/DISPLAY` commands is suitable for this level of monitoring, together with end-user feedback. One use of the Resource Measurement Facility (RMF) II is to collect information about processor, channel, and I/O device utilization.

The MTO is an important source of information about the behavior of the online IMS system. An important part of MTO feedback is the set of conditions during an IMS Monitor run. This information can help establish the validity of the monitor data.

With the status information that can be obtained using the `/DISPLAY` command, you can arrange to get a processing status during online execution. The status can include the queue levels, active regions, active terminals, and the number and type of conversational transactions. Such a status can be obtained with the aid of an automated operator program invoked by the MTO. At prearranged milestones in the production cycle—such as before scheduling a message or BMP region, at shutdown of part of the network, or at peak loading—the transaction processing status and measures of system resource levels can be recorded.

Daily Monitoring

Measure and record key system parameters daily. Record both the daily average and the peak period (usually one hour) average. Compare against major performance objectives and look for adverse trends.

This data usually consists of counts of events and gross-level timings. In some cases, the timings are averaged for the entire IMS system, for example, elapsed times for input queuing or program execution.

You can use the IMS system log as input to offline processing to produce statistics on a daily or regular basis. Two utilities, IMS Log Transaction Analysis and IMS Statistical Analysis, are suitable for this level of monitoring, because they impose no additional processing load on the online system.

Detailed Monitoring

Periodically collect detailed statistics on system operation for performance analysis against system-oriented objectives and workload profiles.

Data at this level is much more voluminous. It typically contains sequences of events and tabulations. The timings reported are at a detailed level.

At this level of monitoring, special trace tools such as the IMS Monitor and Generalized Trace Facility (GTF) are useful. They collect a detailed sample of the online processing and distinguish between activity in dependent regions, asynchronous processing for terminals and message queues, buffer pool usage, and system data set I/O.

Additional information on using these monitoring tools is included later in this chapter. The use of monitoring and tools to detect performance problems is explained in “Identifying and Correcting Performance Problems” on page 284.

System for Generalized Performance Analysis Reporting (GPAR), program number 5798-CPR, is designed as a base for reporting programs (IBM or user-written). It helps summarize sequential activity traces like the IMS system log, IMS Monitor tapes, and GTF traces. It also contains facilities to print user-tailored graphs from any performance data log or non-VSAM sequential data set. GPAR is a prerequisite for ASAP II, IMSPARS, VTAMPARS, and GTFPARS.

Monitoring Multiple Systems

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

You should plan to obtain both statistical and performance data for IMS online systems that are part of a multi-system network. You can use the same monitoring tools that are used for generating performance data for single IMS systems:

- The IMS Monitor can be executed concurrently in several systems. You obtain IMS Monitor reports for each individual IMS system and coordinate your processing analysis.
- The IMS Statistical Analysis utility produces summaries of transaction traffic for each individual system. Again, you combine the statistics for a composite picture.
- The IMS Transaction Analysis utility enables you to trace transactions across multiple systems and examine the traffic using the various active physical links.

Coordinating Performance Information

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The IMS system log for each system participating in Multiple Systems Coupling (MSC) contains only the record of events that take place in that system. The logging of traffic received on links is included. You can augment the system log documentation that records the checkpoint intervals with the system identifications of all coupled systems. This helps you interpret reports, because you know of transactions that might be present in message queues but are not processed, and you can expect additional transaction loads from remote sources. In your analysis procedures, include ways of isolating the processing triggered by transactions originating from another system.

To satisfy the need for monitoring with typical activity that includes cross-system processing, coordinate your scheduling of the IMS Monitor and other traces between master terminal operators. The span of the monitoring does not have to be

exactly the same, but if it is widely different, the averaging of report summaries can make it harder to interpret the effect of the processing triggered by cross-system messages.

Related Reading: For more information on interpreting MSC reports, see *IMS/ESA Utilities Reference: Transaction Manager*.

Monitoring Fast Path Systems

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The major emphasis for monitoring IMS online systems that include message-driven Fast Path application programs is the balance between rapid response and high transaction rates. With Fast Path, performance data is made part of the system log information. Because the bulk of the online traffic is expected to be handled by expedited message handling and not through the message queues, the IMS Monitor is not the prime tool for monitoring Fast Path application programs. You can use the IMS Fast Path Log Analysis utility to generate statistical reports from the system log records. This utility can provide summaries of the Fast Path transaction loads, reports that highlight exceptional response time, and a breakdown of the elapsed time between key events during the time in the system.

The system administration tasks of setting up a monitoring strategy, performance profiles, and analysis procedures should be carried into the Fast Path environment.

Related Reading: For more information on using the IMS Fast Path Log Analysis utility, see *IMS/ESA Utilities Reference: System*.

Transaction Flow

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

A distinct sequence of events occurs during the processing of a transaction. Message-related processing is asynchronous within IMS, that is, not associated with a dependent region's processing. Examples of this kind of processing are message traffic, editing, formatting, and recovery-related message enqueuing, any of which can be done concurrently with application program processing for other transactions. Events from application program scheduling to termination are associated with a PST and can be regarded as synchronous.

To get an overall picture of the activity that takes place when an online IMS system is processing a mix of transactions concurrently, see Figure 41 on page 255. The figure shows a sequence of events. Each event is explained in notes that follow the figure.

The unit of work by which most IMS systems are measured is the transaction (or a single conversation iteration, from entering of an input message to receipt of one or more output messages in response). One way of representing the flow of units of work is to compare it to three funnels through which all transactions must pass, as illustrated in Figure 41 on page 255. The events that account for the principal contributions to transaction response time are numbered in the center. The items entered on the left of the diagram are message related, and those on the right are related to the application program. The arrows trace the flow for an individual

transaction. (The diagram does not show the paging element or system checkpoint processing that is distributed through the elapsed times.)

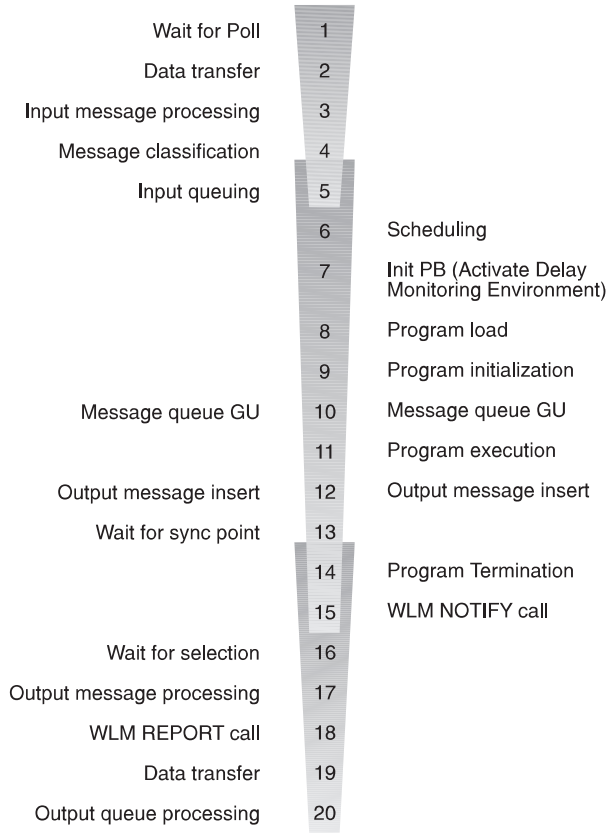


Figure 41. Processing Events during Transaction Flow through IMS

Notes to Figure 41:

1. Wait for Poll.

   This is the time between pressing the Enter key and receiving a poll that results in the data being read by the channel program. With several control units on a line, a slight delay is probable, because each control unit is polled in turn until one is ready to send. There can also be delays caused by data being transmitted on the line from (or to) another terminal. Also, IMS communications can be currently processing an input message from a terminal on the line and polling has not recommenced. Some form of hardware monitor is required to measure the time waiting for poll.

2. Data Transfer.

   This time includes propagation delay and modem turnarounds for multi-block input messages. You can estimate the data transfer times if the volume of data transmitted is known.

3. Input Message Processing.

   The IMS control of the transaction begins when the input message is available in the HIOP. The time the message spends in this pool, in MFS processing, and in being moved to the message queue buffers affects response time. Individual transaction I/O to the Format library affects the message queue. A major factor in determining response time is whether the respective pools are

large enough for the current volume of transactions flowing into input queuing. In particular, if the message queue pool is too small, overflow to the message queue data sets occurs.

4. Message Classification.

   This is the call to the MVS WLM to obtain a WLM service classification for the incoming message.

5. Input Queuing.

   This is the time spent on the input queue or in the message queue buffers waiting for a message region to become available. In a busy system, this time can become a major portion of the response time. The pattern of programs scheduled into available regions and the region occupancy percentage are important and should be closely monitored.

6. Scheduling.

   Because of class scheduling, regions can be idle while transactions are still on the queue. The effects of scheduling parameters can be:

   - Termination of scheduling as a result of PSB conflict or message class priorities
   - Termination of scheduling as a result of intent conflict
   - Extension of scheduling by I/Os to IMS.ACBLIB for intent lists, PSBs, or DMBs
   - Pool space failures in either the PSB or DMB pools

7. Init PB Call (Activate Delay Monitoring Environment).

   Activate the WLM delay monitoring environment for the message when it is placed into the dependent region. The WLM PB is initialized with the Service Classification and transaction name, message arrival time, program execution start time (current time), userid, and so forth.

8. Program Load.

   See event 9.

9. Program Initialization.

   After scheduling, several kinds of processing events occur before the application program can start:

   - Contents supervision for the dependent region
   - Location of program libraries and directories to them
   - Program fetch from the program library
   - Program initialization up to the time of the first DL/I call to the message queue

   For monitoring, you can obtain the overall time for the above activities. The number of I/Os should be checked periodically.

10. Message Queue GU.

    This is the GU call to the message queue. It is chosen as a measuring point because the event is recorded on the system log and is used as a starting point for iterations of processing when more than one message is serviced at a single scheduling of the program.

11. Program Execution.

    The time for program execution, from first message call to the output message insert, is a basic statistic for each individual transaction. It is important to account for that time in terms of the work performed:

    - The number of transactions processed per schedule
    - The number and type of DL/I calls per transaction

- The number of I/Os per transaction

A useful breakdown of elapsed time into processor time and I/O helps determine which transactions use significant resource.

12. Output Message Insert.

    This time begins the asynchronous processing for the output response. The output message requests flow into the funnel to be serviced while the application program is either beginning to process another input message or is performing closeout processing and program termination.

13. Wait for Sync Point.

    When an output message is issued by a program, it is enqueued on a temporary destination until the program reaches a synchronization point. For programs specified as MODE=MULT, a long delay in output transmission can occur when the program goes on to process several transactions at one scheduling. None of the previous output messages can be released for transmission until the program ends. If the program fails, all current transactions are backed out. Actually, when the messages are dequeued, LIFO sequence is used, from temporary to permanent destination. With MODE=SNGL, the wait for sync point (at the next GU to the message queue) is normally negligible.

14. Program Termination.

    In the case of MODE=MULT, discussed in event 13, the synchronization point occurs at program termination. Any database updates are purged from the database buffer pools, and the waiting output messages are released.

    In the MODE=SNGL case, the synchronization point occurs at the previous message queue GU call (usually a GU with a QC status code), and no database commit processing occurs at termination, unless the application program has updated a database after the last message queue GU.

15. WLM Notify Call.

    This tells WLM that the application program has ended execution. The PB and current time are passed to WLM.

16. Wait for Selection.

    This time is similar to Wait for Poll on input, with one difference—an output message might not have to wait for the completion of a polling cycle if no polling is in progress on the line at the time the output message is enqueued. However, there might be a wait for the duration of data transmission to other terminals on the line. In a busy system, this could account for the majority of time spent on the output queue.

17. Output Message Processing.

    This activity is similar to event 3 on page 255.

18. WLM Report Call.

    This tells WLM the response is being sent. IMS passes the input message arrival time, the Service Classification, the current time (output message send time).

19. Data Transfer.

    This activity is similar to event 2 on page 255.

20. Output Queue Processing.

    Output messages that have been sent are dequeued after their receipt has been acknowledged by the terminal. In the case of paged output, the acknowledgment is a consequence of another input or a PA2 entry from the terminal.

The IMS Monitor

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The principal monitoring tool provided by IMS is the IMS Monitor. It is an integral part of the control program in the DB/DC environment. (The counterpart of the IMS Monitor in the batch environment is the Database Batch Monitor.)

The IMS Monitor collects data while the DB/DC environment is operating. Information in the form of monitor records is gathered for all dispatch events and placed in a sequential data set. The IMS Monitor data set is specified on the IMSMON DD statement in the control region JCL; data is added to the data set when the `/TRACE` command activates the monitor. The MTO can start and stop the monitor, guided by awareness of the system's status, to obtain several snapshots.

Related Reading: For more information on interpreting IMS Monitor reports, see *IMS/ESA Utilities Reference: System*.

DBCTL Considerations

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

This section explains how to establish monitoring procedures for your DBCTL environment. To understand monitoring in a DBCTL environment requires a comparison of DBCTL and DB/DC. In a DB/DC environment, monitoring generally refers to the monitoring of transactions. The transaction is entered by an end user on a terminal, is processed by the DB/DC environment, and returns a result to the user. Transaction characteristics that are measured include total response time and the number and duration of resource contentions.

A DBCTL environment has no transactions and no terminal end users. It does, however, do work on behalf of CCTL transactions that are entered by CCTL terminal users. DBCTL monitoring provides data about the DBCTL processing that occurs when a CCTL transaction accesses databases in a DBCTL environment. This access is provided by the CCTL making the DRA request.

The most typical sequence of DRA requests that represents a CCTL transaction would be:

- A SCHED request to schedule a PSB in the DBCTL environment
- A DL/I request to make database calls
- A sync point request, COMMTERM, to commit the database updates and release the PSB

The DBCTL process that encompasses this request is called a unit of recovery (UOR).

DBCTL provides UOR monitoring data, such as:

- Total time the UOR exists
- Wait time to schedule a PSB
- I/O activity during database calls

This information is similar to, and is often the same as, DB/DC monitoring data. However, in a DBCTL environment, the UOR data represents only a part of the total processing of a CCTL transaction. You must also include CCTL monitoring data to get an overall view of the CCTL transaction performance.

In this section, the term ″transaction″ refers to a CCTL transaction. When it applies, UOR is specifically named.

Establishing Monitoring Procedures

The CCTL administrator must decide what strategy to use to monitor transaction performance. This section describes some possible strategies and how IMS functions can help.

Several types of monitoring strategies are available. You can:

- Summarize actual workload for the whole of the online execution. This can be continuous or at an agreed-to frequency. Total workload or selected representative transactions can be tracked.
- Take sample snapshots at peak loads and under normal conditions. It is always useful to monitor the peak periods for two reasons:
  - Bottlenecks and response time problems are more pronounced at peak volumes.
  - The current peak load is a good indicator of what the future average will be like.
- Monitor critical transactions or programs that have documented performance criteria.

Plan your monitoring procedures in advance. A procedure should explain the tools to be used, the analysis techniques to be used, the operational extent of those activities, and how often they are to be performed.

You need to:

- Develop performance criteria
- Develop a master plan for monitoring, data gathering, and analysis

The sections that follow summarize the performance administration activities for these two tasks.

Establishing Performance Objectives

Inherent in the design of your online DBCTL/CCTL system are your performance objectives. Establishing performance objectives is a major task requiring data gathering for the DBCTL/CCTL workload as a whole. After defining the workload and estimating the resources required, you must reconcile the desired response with the response you consider to be attainable. Monitor the performance of the system to determine if these objectives are being met.

Base performance objectives on:

- Desired, acceptable, and maximum response time
- Average and maximum resource demands (or workload) per transaction
- Predicted and actual transaction volumes

Establishing your performance objectives is an iterative process involving the following activities:

- Defining CCTL user-oriented performance objectives and priorities

These derive from the way a CCTL end user perceives the service provided by the system. These objectives state expectations of response time as seen by the end user at the terminal.

When response time objectives are established, they should not only reflect the time-in-system (the elapsed time from entry of a last input message segment to the first response segment), but also the expected amount of IMS, CCTL, and application program processing. You should consider whether to define your criteria in terms of the average, the 90th percentile, or even worst-case response time. Your choice depends on your installation's audit controls and the nature of the specific transactions.

- Determining how performance against these objectives is measured and reported to users

  You can combine UOR data with CCTL transaction data for a complete transaction profile, but keep the following things in mind:

  – If a CCTL allows only one UOR per transaction, this one-to-one ratio makes it easy to combine the monitoring data. It is also easier to track, because a CCTL can put a transaction-ID into a SCHED request. DBCTL puts this ID in monitor log records, and it appears in monitor reports.
  – If a CCTL allows for multiple UORs within a single transaction, the total UOR data can be obtained by adding the data from each of the UORs.

  Besides data from the IMS Monitor, UOR data is also provided to a CCTL upon the completion of a request terminates the UOR. This data can be used in either of the cases above. A CCTL can also perform real-time analysis of UOR statistics related to its transactions.

  The same UOR data is also part of the IMS system log, which records the status and end of a UOR.

- Understanding and documenting the current workload

  This requires breaking down the total work into categories and, for each category, developing a workload profile (generally estimates) that include:

  – Definition of the transaction category (for example, transaction type or group of transactions). Two characteristics of the category are:
    - The transaction profile. Most transactions perform a single function and have an identifiable workload profile. Later in the process, transaction types with common profiles can be amalgamated for convenience.
    - The transaction volume. In situations where the workload to be documented is already operational, a summary of transaction volumes can be obtained. In other cases, volumes are estimated.
  – Relative priority of category, including periods during which priority changes
  – Resource requirements of the work:
    - Physical resources managed by the operating system (real storage, DASD I/O, terminal I/O)
    - IMS logical resources managed by the subsystem, such as control blocks, latches, buffers, and number of DRA threads
    - CCTL resources required

  To obtain a base profile of transaction resource demands, you can start DBCTL/CCTL on a dedicated machine and execute a few transactions to accomplish initialization and buffer pool usage. Then start the IMS Monitor and measure a sample of transaction execution.

You can use the base transaction profile to examine the transaction workload to see if there is any way to reduce it. Such design changes result in the greatest impact, occurring before systemwide contention. You can also compare the base profile against the transaction profile in the production environment.

- Translating the resource requirements and volume information obtained into system-oriented objectives for each work category

  This includes statements about the transaction rates to be supported (including any peak periods) and the internal response-time profiles to be achieved.

- Confirming that the system-oriented objectives are reasonable

  After initializing the system and monitoring its operation, you need to find out if the objectives are reasonable (given the hardware available), based upon the measurements of the workload. If the measurements differ greatly from the estimates used, you must revise the workload documentation and system-oriented objectives accordingly, or tune the system.

Deciding on Monitoring Activities and Techniques

When you develop a master plan for monitoring and analyzing performance, you should establish:

- A master schedule of monitoring activity

  Coordinate monitoring with operations procedures to allow feedback of online events to be incorporated into instructions for daily or detailed data gathering.

- Which tools are to be used for monitoring

  The tools used for data gathering should provide for dynamic monitoring, a daily collection of statistics, and more detailed monitoring.

- The kinds of analysis to be performed

  Document what data is to be extracted from the monitoring output, identifying the source and usage of the data. Although the formatted reports provided by the monitoring tools help organize the volume of data, you should probably design worksheets to assist in data extraction and reduction.

- A list of the personnel that are to be included in any review of the findings

  The results and conclusions from analyzing monitor data should be made known to the user liaison group and to system performance specialists.

- A strategy for implementing changes to the DBCTL environment design, and to the CCTL system design, resulting from tuning recommendations

  Coordinate this change implementation strategy with your installation's standards for testing and standards for frequency of production environment changes. Change management is described in "Chapter 9. Modifying Your System Design" on page 295.

Plan for three broad levels of monitoring activity:

- Dynamic
- Daily
- Detailed

Dynamic Monitoring

Observe the system's operation continuously to discover any serious short-term deviation from performance objectives.

This section refers to IMS functions and general concepts related to dynamic monitoring. If the CCTL has similar functions (for example `DISPLAY` commands), they should also be used.

DBCTL Monitoring Considerations

The output from /DISPLAY commands is suitable for this level of monitoring, together with end-user feedback. One use of the Resource Measurement Facility (RMF) II is to collect information about processor, channel, and I/O device utilization.

The MTO is an important source of information about the behavior of the online IMS system. An important part of MTO feedback is the set of conditions during an IMS Monitor run. This information can help establish the validity of the monitor data.

With the status information that can be obtained using the /DISPLAY command, you can arrange to get a processing status during online execution. The status can include the pool levels and active regions. At prearranged milestones in the production cycle—such as before scheduling a message or BMP region, at shutdown of part of the network, or at peak loading—the transaction processing status and measures of system resource levels can be recorded.

Daily Monitoring

Measure and record key system parameters daily. Record both the daily average and the peak period (usually one hour) average. Compare against major performance objectives and look for adverse trends.

The IMS system log can be used as input to offline processing to produce statistics on a daily or regular basis. The 07 and 08 log records contain the UOR data.

Detailed Monitoring

Periodically collect detailed statistics on system operation for performance analysis against system-oriented objectives and workload profiles.

Data at this level is much more voluminous. It typically contains sequences of events and tabulations. The timings reported are at a detailed level.

At this level of monitoring, special trace tools such as the IMS Monitor and Generalized Trace Facility (GTF) are useful. They collect a detailed sample of the online processing and distinguish between activity in CCTL threads, buffer pool usage, and system data set I/O.

Related Reading:

- For information on using the IMS Monitor, see *IMS/ESA Utilities Reference: System*.
- For information on how monitoring and tools are used to detect performance problems, see “Identifying and Correcting Performance Problems” on page 284.

System for Generalized Performance Analysis Reporting (GPAR), program number 5798-CPR, is designed as a base for reporting programs (IBM or user-written). It helps summarize sequential activity traces like the IMS system log, IMS Monitor tapes, and GTF traces. It also contains facilities to print user-tailored graphs from any performance data log or non-VSAM sequential data set. GPAR is a prerequisite for ASAP II, VTAMPARS, and GTFPARS.

The IMS Monitor

The principal monitoring tool provided by IMS is the IMS Monitor. It is an integral part of the control program in the DBCTL environment. (The counterpart of the IMS Monitor in the batch environment is the Database Batch Monitor.)

The IMS Monitor collects data while the DBCTL environment is operating. Information in the form of monitor records is gathered for all dispatch events and

placed in a sequential data set. The IMS Monitor data set is specified on the IMSMON DD statement in the control region JCL; data is added to the data set when the /TRACE command activates the monitor. The MTO can start and stop the monitor, guided by awareness of the system's status, to obtain several snapshots.

Related Reading: For more information on interpreting IMS Monitor reports, see *IMS/ESA Utilities Reference: System*.

Chapter 8. Tuning Your System

You can make changes to parameters that allow the IMS online system to execute more efficiently. Specifically, these changes to parameters allow IMS to:

- Attain performance criteria
- Efficiently use resources

In this Chapter:

"Chapter 7. Monitoring Your System" on page 247 explains setting up performance objectives and monitoring, which are prerequisites to tuning activities for the IMS online system. Terms defined in that chapter are not generally explained again. Your data for analysis probably depends on the reporting capabilities of one or more of the following tools:

- The Log Transaction Analysis utility
- The Statistical Analysis utility
- The IMS Monitor
- Report programs executing under GPAR: IMSASAP II
- Resource Monitoring Facility (RMF II)

This chapter presents a general strategy for tuning the IMS online system. After some explanation of performance management, the chapter approaches tuning in two stages:

- Initializing SCP (system control program) and IMS parameters
- Identifying and correcting performance problems

For each activity, several major areas of concern are identified. You should examine the potential tuning activities in each area. This is an outline strategy and not a substitute for more formal performance studies. However, this information should prepare you to address most major problems.

Related Reading: For more information on IMS online systems that use Multiple Systems Coupling or Fast Path, or that participate in data sharing, see:

- *IMS/ESA Operations Guide*
- *IMS/ESA Utilities Reference: System*
- *IMS/ESA Utilities Reference: Transaction Manager*

Managing Performance

IMS performance management is made up of a set of activities, designed to achieve the following goals:

1. Establish performance objectives for each subsystem (IMS, TSO, Batch) and their relative priorities.
2. Initialize SCP and subsystem parameters with a view to meeting the defined performance objectives.

3. Monitor the operation of the total system to confirm that performance objectives are being met.
4. Identify a problem that has been monitored and correct it with minimum disruption of normal service to users.
5. Perform capacity planning or trend analysis to ensure that projected loads will not cause performance problems in the foreseeable future.

Items 1 on page 265 and 3 are addressed in “Chapter 7. Monitoring Your System” on page 247. Item 5 is described in *IMS/ESA Operations Guide*.

This chapter describes items 2 on page 265 and 4 as they relate to tuning. These two items can be characterized as guidance to reduce the chance of initial problems and a methodology for ongoing problem resolution. However, you should begin by understanding how they relate to the other aspects of the performance management process.

Tuning actions and system redefinition are two aspects of IMS performance management that are part of a continuous cycle of activities, as illustrated in Figure 42. Interpreting system performance includes both design decisions and prediction.

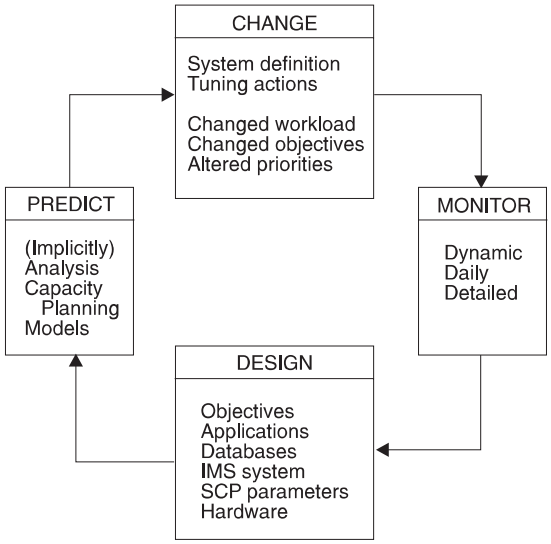


Figure 42. Activity Cycle for IMS Performance Management

Change Management

Performance management is really the management of change. Performance is stable if few changes occur in an IMS system. More common is a continually changing environment where the IMS system must be adjusted to the addition of new application programs and respond to changes in:

- Application design
- Transaction volumes
- Database volumes
- Workload mix
- User population
- User priorities
- Volumes of non-IMS work

- Hardware configuration

IMS tuning attempts to allocate resources for efficient transaction processing.

You can use various techniques to tune the system during operation. For example, you can:

- Start new message regions
- Change transaction PARLIM, PROCLIM, or class assignments
- Alter message region class assignments

These activities involve ensuring that sufficient IMS resources, such as regions, pool space, or control block areas, are provided. The IMS configuration needs to be matched to the installation's physical resource constraints. Typically, the use of virtual storage is exchanged for some I/O activity controlled by IMS, and contention is reduced for data stored on DASD devices. Tuning should not be crisis management but a continual compensation for changes or workload trends.

A systematic approach to tuning is included in "Identifying and Correcting Performance Problems" on page 284.

The Levels of Monitoring

Three broad levels of monitoring are described in "Chapter 7. Monitoring Your System" on page 247:

- Dynamic monitoring to discover if performance deviates from established objectives
- Daily monitoring of key performance objectives in order to observe trends and early indicators of possible performance problems
- Detailed monitoring performed periodically to review performance and decide on tuning requirements

Design Variables

The performance of an IMS system depends on the interaction of the workload with many design variables. Performance is affected by:

- IMS system definition
- IMS execution-time options and parameters
- Application design
- Database design
- Operating system configuration
- Hardware configuration

These areas are addressed in "Initializing SCP and IMS Parameters" on page 268. Design aspects also include allowing for design and tuning changes decided upon as a result of monitoring the system in operation.

Types of Prediction

Two aspects of prediction exist: implicit and explicit prediction.

Implicit

When an alteration, classified as a tuning change, is made to the online IMS design, a performance improvement is implied.

Explicit
When actions are taken to make predictions, analytical or simulation models are used, and capacity planning is performed.

Initializing SCP and IMS Parameters

Initializing SCP and IMS parameters involves the activities explained in the following sections:

The IMS and SCP parameter initialization steps taken reflect the actions that can have a performance impact. These parameters must be set while considering other aspects. For example, if design or function requirements that contradict or override these recommendations exist, you must evaluate the alternatives.

Assigning SCP Dispatching Priorities

Choose a coordinated dispatching priority scheme for the total system. The scheme must reflect the relative worth of any subsystems that are to execute concurrently. The IMS control region operates in protect key 7 as a system task. This reduces the number of executable instructions for some supervisory functions.

Ordering Dispatching Priority

The job dispatching priority dictates the sequence jobs are given to available machine cycles. Any job with a dispatching priority higher than the control region causes interference. Although VTAM or TCAM normally runs at high priority (as do JES and the Master Scheduler for MVS), you should review the position of other work (principally TSO and batch) in relation to the IMS control region and the dependent regions.

Setting Dispatching Priority for Dependent Regions

It is important that both the control region and the dependent regions obtain priority processing. The parallel DL/I function in IMS enables most of the DL/I call processing to be performed in the dependent regions. If the assigned dispatching priority is too low, the promptness of service to application programs declines. Usually, you set the BMP region priority lower than MPP priorities. The BMP priorities should, however, immediately follow the MPPs, because they contend for the same control region resources, such as program isolation enqueues. Normally, TSO "first period" work is placed below the IMS control region; it might be above the dependent regions. TSO "second period" work and batch are normally placed below the IMS dependent regions.

Setting a Performance Group Number under MVS

You can assign a unique performance group number to each IMS region. In this way, you can obtain RMF report statistics separately for each region.

If the DL/I address space option is used, you might decide to include both the control and DL/I address spaces in the same RMF performance group.

Setting Priorities in a DBCTL Environment

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
		X	

For a DBCTL environment, the CCTL region should be higher priority than DBCTL, which in turn, should be higher priority than any batch processing regions.

Choosing IMS Options for Performance

Performance implications are associated with many parameters. Some parameters are chosen for system definition, some for execution-time options, and some are an integral part of design decisions. The sections that follow highlight many of the options.

Optimizing IMS and Application Module Loading

Where possible, unless availability of virtual storage is a constraint, use the pageable link pack area (PLPA) for IMS reentrant modules, region and program controllers, frequently used high-level language modules, any reentrant application code, and the overlay supervisor (if used).

Use the execution-time parameter SRCH=1 in the IMS procedure to cause the job pack area and link pack area to be searched before STEPLIB or JOBLIB.

Program Library Considerations

Place a program library containing the production programs first in the STEPLIB concatenation for the message regions. The sequence of programs in IMS.PGMLIB should be in descending order based on frequency of use.

The order of the system search for program libraries can account for some inefficiency in program load. STEPLIB or JOBLIB is the first program library searched. If one of these is not used, IMS.PGMLIB should be first in the LNKLSTnn member of SYS1.PARMLIB. The placement of IMS.PGMLIB is far more important than IMS.RESLIB, because its contents are in continual demand for application program scheduling rather than being in demand only at system initialization. The search and load times are significant because they become a performance penalty each time the program is scheduled. You should block programs to full track size in the library. The DC option of the linkage editor, used for downward compatibility, causes program block sizes to be 1024 bytes. You should link-edit again without this option.

Dependent Region BLDL List

A list of entries is maintained in the dependent region containing the directory index for programs. It is maintained on the most active, most recently used basis, and reduces the I/O to the program directory. Programs with entries in this list have a lower schedule-to-first DL/I call elapsed time than infrequently used programs. You can override the default of 20 entries for the message processing region by using the DBLDL parameter in the EXEC statement of the DFSMPR procedure. The maximum number of entries for the message processing region is 9999.

For a region that is used to test new or changed programs, set the DBLDL parameter to 0, ensuring that the most current version of the program is loaded for each execution. Specifying the DOPT parameter disables quick reschedule.

Application Program Control

Specify MODE=SNGL on the TRANSACT macro statement to reduce message queue I/O activity, and response time.

For the batch message program execution controlled by a time limit, select the IMSBATCH procedure parameter STIMER=2. This causes the STIMER/TTIMER macro to be issued only once per schedule, rather than once per DL/I call with STIMER=1. Similarly, use the value 2 for the positional parameter STIMER for the DFSMPR procedure when controlling message processing regions.

For the IMSBATCH procedure, use parameter values of SPIE=0 and TEST=0 for production programs to eliminate unwanted SVCs. The equivalent positional parameters for the DFSMPR procedure are SPIE and VALCK.

With PL/I, use the latest release of the optimizing compiler to reduce initialization and termination overhead.

Setting Checkpoint Frequency

Adjust the checkpoint frequency so that checkpoints are not taken more frequently than once every 10 to 20 minutes. You control this frequency with the CPLOG keyword on the IMSCTF system definition macro.

Message Format Options

Specify FILL=NULL or FILL=PT in DOFs that have many DFLDs to minimize the transmission of blank characters.

Setting Queuing Options

If you do not want priority processing, you can use the IOS queuing option to set priority processing off. The IOS option causes IMS to process all jobs in the queue on a “first-in, first-out” basis.

Page Fixing IMS Resources

If you want to ensure that the IMS virtual storage is always backed by real storage, you can specify a list of page-fix requests in the member DFSFIXxx in IMS.PROCLIB. The page fix is done during the IMS control region initialization. Long-term page fixing for the queue manager buffers can be requested using the EXVR parameter in the EXEC statement of the control region JCL.

You tune buffer pools to make them large enough to reduce or minimize I/O activity without driving up the system paging rate or constraining virtual storage. Considerations for page fixing buffer pools are given in “Page Fixing the IMS Buffer Pools” on page 278.

Defining IMS Resources for DREF Storage

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

You can request that the IMS virtual storage never migrate to auxiliary storage. By specifying DREF requests in the DFSDRFxx member in IMS.PROCLIB, you ensure that virtual storage does not move beyond expanded storage. If expanded storage is not available, the storage is page-fixed.

Related Reading: For more information on DREF, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

MVS Page Fix Considerations

MVS modules that are frequently referred to by the IMS system should be page fixed at initial program load.

Page fixing certain modules also saves a “Page fix and Wait” SVC for each occurrence of timer facilities. Place these modules close together to minimize the number of pages in the pageable link pack area (PLPA).

Related Reading: For more information on MVS modules, see *OS/VS2 MVS System Programming Library: Initialization and Tuning*.

Avoiding Contention for IMS Resources (Excluding Buffer Pools)

The interleaved processing of concurrent regions is a major area of contention for IMS resources. A set of regions processing a mix of transactions can overlap processing and use the advantage of multi-tasking over serially ordered execution.

The following sections identify several areas concerned with the pattern of scheduling and region efficiency:

Using Class Scheduling

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The following points affect your decisions about using class scheduling:

- Use classes to separate low-priority and long-running transactions from high-priority work.
- Arrange your class structure so that any high-priority class work has the opportunity of being scheduled in more than one region, unless doing so is undesirable for some special reasons, such as program isolation conflicts. Multi-server systems generally produce lower waiting times than single-server systems.
- To prevent message regions from being idle while transactions are waiting on the input queue, assign additional (lower-priority) classes to a message region below the primary class or classes assigned to the region. Avoid regions that only service one class, unless you are sure that work of that class will always be waiting to be processed.
- For very high-volume transactions, dedicate a message region and classify the program as a wait-for-input program to reduce input queuing time and to eliminate scheduling and program load.

Processing Limit Counts

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Review your transaction class and priority scheme. Set the PROCLIM parameter on the TRANSACT macro to ensure that no transaction type can monopolize a message region if other transactions are waiting and are eligible for processing in that region. Only use a PROCLIM value higher than 5 if the response times of contending transactions are unimportant. In that case, set PROCLIM=1 for the lower-priority transactions. With wait-for-input transactions, set PROCLIM to a suitably high (or maximum) value so that you avoid rescheduling the application program. Use pseudo WFI and quick reschedule to eliminate processing overhead. By going through quick reschedule, a region can process more messages per schedule than the processing limit, thereby eliminating unnecessary rescheduling and reloading of application programs.

Note: A region scheduled against a transaction whose processing limit count is 0 cannot go through quick reschedule or become a pseudo WFI.

Parallel Scheduling

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Do not use parallel scheduling for low-volume transactions that can be scheduled sequentially. Using message regions for parallel processing can reduce the efficiency of regions in which multiple transactions are processed for each scheduling of a program. Parallel scheduling should be reserved for those transactions where temporary peaks in the arrival rate might cause excessive queuing to develop due to “flooding” of a single message processing region. In these situations use a PARLIM level geared to the average service time and response time objective for the transaction type.

IMS Message Processing Regions

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Operating your IMS system with too few or too many message processing regions can affect performance and transaction response times. In an environment with many different transaction types, regions with a very high occupancy level must be carefully monitored to ensure that they are not causing queues to build up on the IMS message queues.

In environments with only a few transaction types, high levels of occupancy can usually be sustained, because the application programs can often remain in the region processing multiple transactions for each schedule.

The meaning of high occupancy depends on the transaction mix, transaction characteristics, class scheduling scheme, and response-time objectives. For some transactions, a large queuing delay might be acceptable. For others, a much smaller queuing delay must be achieved in order to satisfy the user.

In contrast, having many low-occupancy concurrent message processing regions can unnecessarily increase the contention among regions for buffer pool space, real storage, IMS system data sets, and database records (because of program isolation enqueuing).

You can eliminate some of the processing overhead by specifying pseudo wait-for-input (pseudo WFI) on the MPP region start-up procedure. A pseudo WFI message processing region remains scheduled when no more messages need to be processed, instead of terminating the application program. This eliminates unnecessary termination and rescheduling.

Adjust the number of message processing regions to the minimum possible consistent with achieving your response-time objectives. Increase the number of regions only when an increasing workload cannot be serviced within the objectives, and when changes to the message class structure, quick reschedule, pseudo WFI, and region control parameters (including PARLIM, PROCLIM, and WFI) do not effectively contain the response-time increases. Although you can specify up to 255 dependent regions, always try to contain the response time demands with a minimum number.

The number of required regions, excluding wait-for-input regions, can be estimated as follows:

$$\#\ \text{regions} = \frac{(\text{arrival-rate}) \times (\text{time-in-region})}{(\text{desired region percentage occupancy})}$$

Arrival-rate is the total number of transactions received in an interval divided by that interval in seconds. *Time-in-region* is defined as the sum of the mean times for: scheduling and termination, schedule-to-first DL/I call, and elapsed execution, taken from the IMS Monitor Region Summary report.

Do not include regions dedicated to wait-for-input processing in this calculation, because their region occupancy is 100%.

In addition to establishing the number of message processing regions, you control which transactions can be scheduled into those regions by:

- The choice of class and priority structure
- The PROCLIM and PARLIM parameters on the TRANSACT macro

CCTL Regions

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

CCTL application program performance can be affected by contention for DRA resources. Specifically, if the maximum number of DRA threads allowed is less than the number of CCTL application programs making PSB schedule requests, some of those requests wait until a DRA thread becomes available.

Related Reading: For more information on CCTL regions, see *IMS/ESA Customization Guide*.

Save Area Sets

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

When you consider the number of regions and their occupancy for a given transaction rate, check that the input data transmission is not delayed. One factor causing delay might be the limiting number of concurrent terminal I/Os. You can adjust this limiting number using the SAV parameter in the online EXEC statement (up to a maximum of 999 concurrent I/Os). This parameter is termed the number of "dynamic save area sets" for the terminal I/O, because one of these areas controls each active terminal I/O.

Database Contention

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

Avoid databases with a small number of roots or root anchor points. If the databases are accessed by more than one transaction concurrently, program isolation (PI) contention is probable. The use of control records in databases or of records that keep running totals also causes PI enqueuing.

If you cannot avoid databases with a small number of roots, use class scheduling to allocate all such contending transactions to a single region, thus forcing them to process sequentially.

Initial Optimizing of IMS Buffer Pools

Most of the information relating to the operation and tuning of the IMS buffer pools is included in this section. If, after initial optimization of IMS buffer pools, you identify problems in message processing that seem to relate to pool resources, review this section.

This section contains information on:

Message Queue Pool

The message queue pool acts as a buffer to the message queue data sets. If message queue I/O occurs during transaction processing, the message queue data sets can be detected as IWAIT items on the IMS Monitor Region IWAIT report. If the LRECL sizes of the short and long message queue buffers are not in correct proportion to each other, I/O can occur, because the mix of transactions uses up one or the other section of the buffer even though the size of the pool seems large enough.

The number of buffers is specified in the MSGQUEUE macro. You can override the number at execution time with the message queue buffer (QBUF) parameter in the EXEC statement. One tuning technique is to decrease the number of buffers until the monitor data shows increasing IWAIT instances marked as message queue I/O. Then you can use the QBUF parameter value plus one page-size unit.

You might want to identify unusual demands on message queue handling. Such demands can include:

- Large multi-segment input and output messages
- Extended processing between segment inserts
- Remote print output
- Terminal operator paging
- Multi-terminal output
- Excessive numbers of program-to-program switches
- Zero-priority messages for BMPs
- SPA sizes for conversational transactions

SPA sizes can adversely affect the performance of the message queues, detracting from the efficiency of single-segment messages that are used as input to well-designed application program processing.

The message queue LRECL size must allow for the SPA size, because the SPA is placed on the queue as the first segment of the message. The SPA segment can be split into multiple logical records when it is written to the IMS.LGMSG data set, or a larger LRECL sizes can be specified. This choice depends on the relative number of transactions requiring large SPA sizes.

You can use the Log Transaction Analysis utility to monitor the I/O queue times. The IMS Statistical Analysis utility produces a Line and Terminal report showing the average size sent and received, but does not show separate long and short message statistics. Average message lengths and counts for each transaction and terminal are part of the information in the system log type X'40' checkpoint records.

Message Format Pool

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Allow for a number of format block pairs (I/O) to be available at one time for transaction processing. To obtain storage in the message format pool, the space holding the least recently referenced format block is freed so that blocks for transactions with high arrival rates are usually available. Plan to have sufficient space in the pool to hold the input and output format block pairs for use by your most frequently used transactions, in order to minimize format block I/O.

Using fetch request elements (FREs) significantly affects efficient use of the pool. Each FRE controls the presence of one format block in the pool. If all the format blocks are 1000 bytes long and 10 FREs are specified, the maximum pool space is 10000 bytes. FRE assignments are made in the BUFPOOLS macro at system definition time but can be overridden by the FRE parameter in the control region EXEC statement. You can use the `/DISPLAY POOL MFP` command to compare the amount of free space to the amount of total space. If the free space is consistently high, relative to the size of the pool, make sure that the FRE assignment is not preventing full use of the allocated storage.

Another way to improve the use of the message format pool is to make it possible for application programs to share formats. For example, a generalized message input format can share common DIF/MID blocks with subsequent editing performed by Field Edit and Segment Edit exit routines.

A performance-related function is available with MFS. The MFS utility enables you to generate a directory of direct pointers to the location of all or a selected subset of the format blocks. This index is made resident in the message format pool and minimizes the I/O required to look up the location of a format block in the active IMS.FORMATA/B library directory. (Each index entry requires 14 bytes.) Evaluate the trade-off between allocating index storage for at least the highly referenced blocks against the reduction of I/Os per block.

Related Reading: For more information on MFS utility processing, see *IMS/ESA Utilities Reference: Transaction Manager*.

PSB Pool, PSB Work Pool, and DMB Pool Space Failures

An application program experiences a pool space failure when space in any pool is insufficient to sustain the current number of concurrently executing dependent regions. Each application program needs its PSB, the PSB Work Pool (PSBW), the intent list, and the DMBs to be scheduled. Make all pools large enough to eliminate pool space failures. The general rule for achieving this is to allow sufficient space for the (2R+1) largest blocks that are part of the set DMB, PSB, or PSBW as appropriate, where R is the number of active dependent regions. This allows for fragmentation in the pool space.

PSB Pool and Intent List Considerations

The PSB pool holds all PSBs for scheduled application programs. A PSB remains in the pool until space is needed to load another PSB. Then, if all space is used, the least-referenced inactive PSB is freed. Exceptions to this are those PSBs made resident and not parallel scheduled. These PSBs do not take from the pool allocation and are usually designated for highly referenced (usually preloaded) programs. You can page fix resident PSBs by including DFSPSBRS in the module fix list coded for member DFSFIXxx in IMS.PROCLIB.

If the DL/I address space option is used, two resident PSB pools exist, one in the MVS common area (DFSPSBRS) and one in the DL/I address space (DFSDLIRS).

Similarly, resident-intent lists can be page fixed with the inclusion of DFSINTRS in the DFSFIXxx member. Make intent lists resident.

If a minimum PSB pool size is required, use the formula given in “PSB Pool, PSB Work Pool, and DMB Pool Space Failures”. This results in fewer than one I/O per scheduling, but no pool space failures, assuming that intent lists are made resident.

DMB Pool Considerations

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

Make the DMB pool large enough to contain all IMS database DMBs. Each time you load a DMB into the pool, the database must be opened, and another database must be closed to free space. This kind of activity increases the elapsed time required for DL/I calls. Limited page faulting is preferable to using OPEN and CLOSE. Make DMBs for active databases resident. These DMBs are good candidates to be page fixed. If all of the DMB pool is not being used, reduce the

size until IWAIT for DMBs begin to occur. The IMS Monitor Region IWAIT report shows the DMB pool size is too small by the presence of DMB= entries. You can then increase the specification of the DMB parameter on the BUFPOOLS macro.

Storage for the PSB Work Pool

A portion of the PSB work pool is required by each active PSB, but the total size need not be larger than the maximum.

When you use the technique of starting large and reducing to some value greater than the maximum amount used, scheduling stops when any pool space failures occur.

Database Buffer Pools

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

Allocate sufficient buffers in the IMS database buffer pools to prevent I/O that results from an application program having to reread data previously brought into the pool. Careful choice of subpool sizes and matching against database block sizes and frequency of database references is required.

Multiple OSAM subpools can be defined as having the same buffer size, and specific data sets can then be directed to specific subpools. If you have many OSAM data sets with similar or equal block sizes, you might be able to obtain some performance advantage by replacing a single large pool with separate subpools. This reduces pool scanning to locate a segment. It can also be used to prevent low-priority transactions (or BMPs) that access many database segments from monopolizing pool space.

Related Reading: For more information on using multiple OSAM subpools, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

You can define multiple VSAM local shared resource pools. If you have many VSAM data sets with similar or equal control interval sizes, you might get a performance advantage by replacing a single large subpool with separate subpools of identically sized buffers. Creating separate subpools of the same size for VSAM data sets offers benefits similar to OSAM multiple subpool support.

Use multiple local shared resource pools to specify multiple VSAM subpools of the same size. Create multiple shared resource pools and then place a VSAM subpool in each one that is the same size as other VSAM subpools in other local shared resource pools. You can then assign a specific database data set to a specific subpool by assigning the data set to a shared resource pool. The data set is directed to a specific subpool within the assigned shared resource pool based on the data set's control interval size.

You can also create separate subpools for VSAM KSDS index and data components within a VSAM local shared resource pool. Using these subpools can be advantageous, because index and data components do not need to share buffers or compete for buffers in the same subpool.

Multiple VSAM local shared resource pools enhance the benefits provided by hiperspace buffering. Use hiperspace buffering to allocate more buffers of 4 KB and multiples of 4 KB by using expanded storage in addition to virtual storage. Using

multiple local shared resource pools and hiperspace buffering allows data sets with certain reference patterns (for example, a primary index data set) to be isolated to a subpool backed by hiperspace, which reduces the VSAM read I/O activity needed for database processing.

Related Reading: For more information on coding the IMS control statements required to activate hiperspace and multiple VSAM local shared resource pools, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Using Sequential Buffering

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

Elapsed time required to sequentially process OSAM databases can be reduced considerably by using Sequential Buffering (SB).

Related Reading: For more information on how SB works and the benefits of using it, see *IMS/ESA Administration Guide: Database Manager*.

Page Fixing the IMS Buffer Pools

Information for initial optimization of IMS buffer pools is explained in preceding sections of this chapter. Apply these guidelines to your system to achieve a minimum value for the total I/O operations (pool I/O and paging I/O). This means that you must consider a trade-off between I/O and storage across all pools, as well as within a single pool. For example, halving the maximum size of a large PSB pool might result in a very small increase in I/Os, but giving that extra storage to the message queue or format pools might result in reducing even more I/O saving.

The way you tune your pools depends upon your real storage environment. In general, increasing maximum pool sizes increases paging; reducing pool sizes reduces paging of those areas.

If the IMS pools are subject to paging and IMS is sharing the processor usage, you can consider page fixing the significant buffers necessary for message and database access. This can be useful in a non-dedicated IMS system with very low-transaction rate, where IMS might be paged out unless the pools are page fixed. In heavily loaded IMS systems, response times might benefit from page fixing the data communication pools, because they hold data over the longest periods.

The principal IMS pools that can be page fixed are:

- Message queue pool (QBUF)
- DMB pool (DLDP)
- PSB pool (DLMP)
- Message format pool (MFBP)
- Database subpools

If the DL/I address space option is used, two PSB pools exist: DLMP in the MVS common area and DPSB in the DL/I address space.

Tell IMS to page fix all these areas, except the database subpools, by naming the resource in control statements that are included in the DFSFIXxx member of IMS.PROCLIB. The corresponding names used in these control statements are shown in parentheses in the above list.

To page fix the database subpools, you use control statements in the DFSVSMxx member of IMS.PROCLIB. The IOBF statement controls OSAM subpools. VSAM pool page fixing is specified with the VSAMFIX keyword of the OPTIONS control statement.

Related Reading: For more information on specifications for the DFSFIXxx and DFSVSMxx members, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Usually, I/O for a pool impacts performance less than page faulting through the pool. If storage is a constraint, using a smaller fixed pool is preferable to using a larger pool that is subject to paging. However, page fixing is unlikely to improve performance, and might lead to higher overall paging rates, if your system is dedicated to IMS. This is because page fixing leaves a smaller available page pool for the rest of the system to use. Unless some other action is taken to reduce the storage requirements, higher rates inevitably result.

Trade-offs Between I/O Controlled by IMS and Paging

When you are choosing between I/O controlled by IMS and paging, you should consider several trade-offs, which are explained in this section.

General Buffer Pool Considerations

As explained in “Page Fixing the IMS Buffer Pools” on page 278, small tuned pools in IMS tend to minimize paging. In addition, because paging is not controlled by the IMS subsystem, it can escalate dramatically in a storage-constrained environment. In contrast, IMS buffer pool I/O is controlled and can be estimated and measured against those estimates. Finally, using a page fault and page I/O operation is generally more costly in processor terms than using a BSAM or IMS OSAM I/O.

Therefore, optimization of IMS in this area requires adjusting pool sizes while keeping in mind the costs in terms of paging and path length of scanning a pool.

IMS Log Buffers

Similar concepts presented in the previous section apply when selecting the IMS online log data set (OLDS) block size and number of buffers in order to minimize system log I/O. The log buffers only need to be plentiful enough to prevent IMS from waiting for a log write in order to get buffer space. The write-ahead data set (WADS) ensures that only full log buffers are written to the OLDS. The log write-ahead function does not truncate log buffers.

Some buffer space is used for dynamic backout. When the backout records exist in the output buffer, they are used directly from the buffers. Otherwise, a buffer is removed from the set of output buffers and used to read the required log block from the OLDS. When backout is complete, the buffer is returned for use as an output buffer. Log buffers are long-term page-fixed.

Using Virtual Fetch under MVS

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

As an alternative to IMS program loading with MVS Program Fetch, you can specify the use of Virtual Fetch. You specify names of the modules to be managed by Virtual Fetch in member DFSVFLxx in IMS.PROCLIB. When the message processing region is initiated, you specify the suffix xx as a parameter (VSFX) in the

EXEC statement of the DFSMPR procedure. Programs managed by Virtual Fetch might have a schedule-to-first DL/I call elapsed time less than those that use Program Fetch, depending upon the performance of the MVS paging subsystem. If your installation has sufficient virtual storage and real storage available, the VFREE parameter on the EXEC statement can use the default value N. This causes all programs that execute in the message processing region to be assigned unique areas of virtual storage. This option can be selected for a high-priority region, for example, one that is scheduled for only a few transaction types. In storage-constrained environments, however, specify the value Y for the VFREE parameter so that IMS releases virtual storage held for that program.

The MVS service Library Lookaside (LLA) is recommended over Virtual Fetch.

Related Reading: For more information on LLA, see "Using Library Lookaside under MVS".

Using Library Lookaside under MVS

As an alternative to IMS program loading with MVS Program Fetch or Virtual Fetch, you can specify the use of the MVS service Library Lookaside (LLA). LLA is the recommended method for managing dynamically loaded program libraries. For IMS application programs, it saves load modules in a data space, so the modules can be retrieved more efficiently than through DASD. Virtual Fetch also performs this function. LLA, however, is more efficient, because it monitors the frequency with which modules are accessed and readjusts its data space population to optimize the probability of having a module in the data space when it is requested.

Guidelines for Application Program Preload

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

The preload option is commonly used for high-activity programs that do not use a large amount of virtual storage. You specify the names of the modules to be preloaded in member DFSMPLxx in IMS.PROCLIB. When the message processing region is initiated, you specify the suffix xx as a parameter in the EXEC statement of the DFSMPR procedure. Preloaded application programs are then branched to directly rather than through a FETCH program. Preloaded programs should have a schedule-to-first DL/I call elapsed time less than those that use the FETCH program, but page fault serialization can cause the application program elapsed time to increase.

Guidelines for the most effective implementation of program preload are:

- All commonly used PL/I, VS Pascal, or COBOL subroutines and application program subroutines should be preloaded into each dependent region. If these subroutines are reentrant, they should be put into the pageable link pack area (PLPA) so that only one copy resides in real storage. Match subroutine usage with the region preload lists to ensure that the appropriate modules are preloaded.
- Application programs are candidates for preload when they account for a high percentage of a region transaction volume. Preload is most effective when the transaction arrival rate for the preloaded program is adequate to keep the working set of the program in real storage. If a system is constrained by real storage contention, preload only subroutines and very high-volume application programs, because preloading can increase the paging rate.

In addition to deciding to preload them, consider class scheduling of high-volume transactions. Depending on the transaction arrival rate, it can be advantageous to preload in only one or two regions and class schedule the transactions accordingly.

The use of the preload option for system performance improvements is highly dependent on the availability of real storage and the arrival-rate of the candidate transactions.

- Application program overlay is sometimes a viable alternative to preload. If an all-purpose application program is coded to interrogate the input transaction data and execute only a portion of the application program code for the transaction subcode, program overlay I/O might be more efficient than loading or paging through the entire program. Preload is a better alternative if the transaction arrival rate is sufficient to maintain a working set in main storage.

Minimizing Path Length

The total number of machine instructions that are executed to process a transaction, including system services, IMS services, and the application program itself, has a direct bearing on throughput. The accumulation of executed instructions is termed the *path length*. The actions suggested in “Choosing IMS Options for Performance” on page 269 all contribute to the minimization of path length. Avoid the regular use of traces such as the DL/I Call Image Capture and other traces invoked by the `/TRACE` command. These are specified as parameters on the OPTIONS statement in the DFSVSMxx member of IMS.PROCLIB.

Do not run the IMS Monitor (DFSMNTR0), except for during 10- to 20-minute preplanned intervals.

In a real storage-constrained system, the most effective way to reduce path length is to minimize paging. Minimal pools help to minimize paging by eliminating costly scanning of directories or buffers that might need to be paged in before they can be read. If virtual storage requirements are reduced:

- A minimal PSB pool minimizes buffer searching.
- A tuned database pool minimizes buffer searching; a larger database pool costs more in path length and might not reduce I/O.
- A tuned message queue pool minimizes buffer searching; a larger pool reduces IMS message queue I/O but does so at the expense of higher processor cycles per queue pool operation.
- The same applies to the message format pool as to the message queue pool.

Considerations for the Communications Network

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Performance factors that affect response time can be related to, and are explained in:

- “NCP and Network Considerations”
- “Host Transaction Processing Considerations” on page 282

NCP and Network Considerations

Review the points in the network where delays might occur and try to avoid bottlenecks. Delays can occur:

- Within a cluster and queue for a line
- Waiting for transmission and waiting to be polled
- Through line errors
- Within the NCP
- Within VTAM
- Waiting for IMS to receive any macro
- Within IMS
- Waiting for transmission of output

Another possible problem is that VTAM cannot easily process larger output transmissions. You should check the appropriateness of the value for the OUTBUF keyword on the TERMINAL macro statements and ETO logon descriptor.

Input can also be delayed because IMS has no more input threads left.

Host Transaction Processing Considerations

Be aware of the trade-off made between the size of VTAM I/O buffers to service the communications network and the use of auxiliary storage. IMS communication traffic might share these resources with TSO and other subsystems.

Use the VTAM authorized path option to reduce processing of SEND and RECEIVE commands by bypassing validity checks of request parameter lists (RPLs) and I/O areas specified in the RPL. You do this by specifying VAUT=1 as a parameter on the EXEC statement for the control region.

IMS transactions should also be specified as response mode where possible. The major benefits are that the number of transmissions and line turnarounds is decreased and that line utilization is reduced.

Guidelines for IMS System Data Set Placement

In addition to tailoring the IMS system data sets to the workload, you should avoid undue contention for their use. Correct placement of high-usage data sets helps you avoid bottlenecks. Here are some guidelines you should consider when placing your IMS online data sets:

- Separate among lightly used DASD volumes: heavily used libraries and data sets such as IMS.PGMLIB, the online log data sets, the active IMS.FORMATA/B and IMS.ACBLIBA/B libraries, and message queues.
- If high-activity data sets are on the same device, place them close to each other. The space allocation should stipulate contiguous storage and allocation by cylinder to avoid fragmentation of stored data.
- Have the volumes available to IMS mounted as private so that operating system temporary data sets are not allocated to these volumes.
- Do not place online IMS system data sets on shared DASD volumes. Avoid contention for both the device and the control unit. Position primary and secondary OLDSs on separate control units and channels when possible.
- To optimize program fetch I/O, order PGMLIB in descending frequency of use. Use full-track blocking. This minimizes Start I/O (SIO) operations and seek times.
- With MVS Virtual Fetch, use multiple local page data sets for each region and separate those local page data sets from VIO and swap data sets.

  Have contiguous space available on the local page data sets when the Virtual Fetch address space is initialized. If no contiguous space is available, device SEEK activity extends program load times.

- If using fixed-head devices, place write-ahead data sets (WADS) and database index data sets on the fixed-head portion.
- If program library contention is a problem, create multiple copies, each for a subset of the message regions. Do not mix production and test program libraries.

I/O Subsystem Configuration

Use the RMF Channel Activity and Direct Access Device Activity reports to observe queuing delays in the I/O subsystem. Configure so that you minimize those delays by balancing the load across physical and logical channels.

Application Optimization

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

If possible, avoid the use of overlays in message processing programs. An exception might be if the application program uses large blocks of code exclusively for part of the transaction processing.

Do not have application programs that perform non-DL/I functions (such as STIMER and TTIMER, and contents supervision). Unless carefully controlled, such application programs usually perform poorly.

If lengthy application programs must be initiated online, consider splitting the application into two parts: an interactive part, which responds to the user and sends a program-to-program message to a batch part, which can be run in a low-priority region.

Consider redesigning applications whose performance is critical. Otherwise, plan to tune the system to process the transaction as it is before embarking on a total rewrite. Consider translating applications whose performance is critical from high-level language to assembler in order to improve program-load and execution time.

Always use the rollback (ROLB) call rather than cause the program to abnormally terminate. This keeps the message region available without operator intervention and with a smaller delay in message region availability to other transactions. (Abnormally terminating an application program keeps the region unavailable to other transactions waiting to be scheduled.)

DL/I Considerations

When designing DL/I databases:

- Use DL/I path calls wherever possible to reduce the DL/I path length and program execution time.
- Use qualified DL/I calls wherever possible to minimize the number of calls required to obtain the desired segment.
- Do not issue a GN call to the I/O PCB if the application program always retrieves single-segment input messages.
- Ensure that the application program returns to issue another GU to the I/O PCB on completion. This allows processing of another transaction of the same type if one exists on the input queue. This avoids the overhead of termination and rescheduling processing.

- When output is sent to several TP-PCBs, define multiple alternate PCBs rather than using the CHNG call.
- Select HDAM as the organization for a database wherever possible, because, when well tuned, HDAM generally results in shorter path lengths and fewer I/Os per DL/I call.

In assessing the characteristics of transaction processing and the interaction of application programs and databases, be aware of the performance design considerations presented in *IMS/ESA Administration Guide: Database Manager* and *IMS/ESA Application Programming: Design Guide*.

Planning for Performance in a Shared-Queues Environment

In a shared-queues environment, individual transaction performance might be poorer than in a non-shared-queues environment. However, because of increased parallelism and workload balancing, overall system performance and throughput should increase. And because input transactions are processed by the originating local IMS system (if a message region is available), you can influence IMS performance by controlling the number of dependent regions for each IMS system.

Each of the following can have an effect on system performance:

- IMS execution parameters: QBUF=, QBUFMAX=, LGMSGSZ=, and SHMSGSZ=
- CQS Local Structure Definition parameter, SYSCHKPT=
- CQS Global Structure Definition parameters: OBJAVGSZ=, OVFLWMAX=, and STRMIN=
- Size of the shared queues structures on the coupling facility
- Size of the MVS system log structure on the coupling facility
- Number of MVS system log data sets to back up the MVS system log structure
- Frequency of structure checkpoints

The size of all log records has increased by eight bytes, and the size of Queue Manager and EMH log records has increased by an additional 32 bytes.

The SPA pool is no longer used, decreasing the amount of data logged for each checkpoint, and increasing the amount of available storage. The SPA is now maintained with its input and output messages.

For Fast Path systems, using the Fast Path Input Edit Router (DBFHAGU0) should be considered carefully because messages defined as “Local Only” will have priority over all other messages (“Local First” and “Global Only”).

Identifying and Correcting Performance Problems

Your general strategy is to decide the origin of the problem being experienced by:

1. Isolating the problem into one of the following categories:
   - High paging rate seems to be slowing some or all of the workload.
   - High utilization of the processor by some subset of the total workload is causing dispatching problems for another (lower-priority) subset.
   - I/O contention in the DASD subsystem is slowing some subset of the workload.
   - Utilization of, or contention for, resources associated with the communications subsystem is slowing the transmission of input or output messages.

- Within IMS, utilization of physical resources (processor, I/O, storage) or contention for logical resources (pools, regions, control blocks, latches) has a negative impact on the performance of some or all of the transactions in one of the following areas:
  - Input or output message processing, including input queuing and message format services
  - Program scheduling and termination
  - Program load and initialization
  - Program execution

2. Investigating further, if necessary, after you isolate the problem area, to determine:
   - The precise nature of the problem
   - The principal offenders by transaction or other category
   - The tuning action most likely to alleviate the problem
3. Taking the appropriate tuning action to prevent the problem from recurring.

Examining Paging Rates

The operating system allocates real storage for the address spaces occupied by the control region or dependent regions. IMS might not obtain adequate machine cycles because of the excessive demand for real storage. Observe the paging rates and the frequency with which processing is delayed by a page fault. You can use the output from RMF II to examine the paging rates at peak IMS loads.

Page faulting in an IMS online system can directly increase the transaction time-in-system by the duration of the paging activity. If page faulting must occur, it is best to confine it to application programs in the dependent regions. A page fault experienced by the control region can hold up any dependent region waiting for a control region service. IMS provides some page-fixing options; these are covered in “Page Fixing the IMS Buffer Pools” on page 278. However, if real storage is constrained, any page fixing generally serves to increase paging elsewhere and is not recommended. The primary tuning technique must be to reduce the virtual and the real storage requirement. This is particularly true in a dedicated IMS environment, where page fixing some portion of IMS only makes paging worse in the unfixed portion, possibly making overall paging rates worse.

Dynamic Monitoring of Paging Rates

You can use the following methods to dynamically monitor paging rates:

Total System Paging Rate
: The RMF Monitor II Paging report gives snapshots of paging activity by sampling interval.

Paging Rate by User
: The RMF Monitor II Address Space Resource Data report gives counts of allocated frames and page faults by address space for each RMF measurement interval. The page fault count is cumulative and includes both local address space page faults (not page I/Os) and CSA/LPA.

If the monitoring indicates that IMS is experiencing temporary delays, it might be possible to stop some TSO or batch initiators.

Daily Monitoring of System Paging Rates

If analysis of RMF Paging Activity reports shows that paging I/O has increased, this must be related to some changes in the workload during the corresponding period. More detailed analysis is usually required to decide on tuning actions.

Detailed Monitoring

You can use the following methods for detailed monitoring:

Paging Rate by User

If analysis at the daily level is insufficient, plan to run a GTF trace. If multiple page data sets are being used, private and global paging can be identified. Examine the GTF Detail Trace report to evaluate the impact on IMS of any paging by calculating the elapsed time delays due to page faults, particularly for the control region. The GTF Page Fault Summary report can be used to discover which area of IMS or system code is being affected by page faults. The report also indicates the type of page faults.

Analyze this type of data to help you decide whether to tune real storage usage. See “Trade-offs Between I/O Controlled by IMS and Paging” on page 279 for other factors to consider.

Examining NOT-IWAIT Time during Scheduling/Termination

The IMS Monitor Region Summary report can help you make an initial assessment of dispatching priority problems by examining the Scheduling or DL/I NOT-IWAIT times, that is, the elapsed time not accounted for by IWAIT time. Increases in the NOT-IWAIT times can be caused by:

- Paging delays
- Dispatching for a higher-priority task

If minimal paging is occurring, the portion of elapsed wait time that occurs during the scheduling and termination of a region is fairly consistent from system to system. This elapsed wait time is not accounted for by IWAIT time. This time is recorded in the IMS Monitor Region Summary report and tabulated under the heading NOT-IWAIT TIME. (DL/I call NOT-IWAIT times can also include dynamic logging I/O delays.) If the total mean NOT-IWAIT TIME is excessive, the machine resource is probably inadequate for IMS. If no higher-priority tasks are present, the cause is probably a high paging rate for IMS scheduler code, control blocks, and PSB pool.

After the reason for the increased storage over commitment is identified, one or all of the following actions must be taken:

- Trade off IMS paging I/O against IMS controlled I/O.
- Reduce IMS paging I/O by reducing over commitment (reduce concurrent users) or by increasing IMS priority relative to other users.
- Speed up paging I/O by reducing I/O contention, splitting page data sets, or by placing page data sets on faster devices.
- In a non-dedicated IMS environment, try to reduce IMS paging, at the expense of other competing work, by fixing some portions of IMS.

Detecting Processor Resource Problems

In an environment with many jobs or subsystems operating concurrently, IMS competes for machine cycles. If those jobs or subsystems, such as an IMS batch execution or TSO, have a higher dispatching priority than the online IMS system, their workload has a marked effect on IMS performance. The processing priority is specified for the IMS regions in the DPRTY parameter on the EXEC statement.

Dynamic and Daily Monitoring of Processor Resource

For dynamic and daily monitoring of processor resources, use the following:

Total System Utilization
The RMF CPU Activity report gives WAIT TIME PERCENTAGE for the RMF report interval, which might typically be 10 to 30 minutes.

Processor Over commitment
RMF Monitor II Real Storage/Processor/SRM report gives percentage processor utilization for each RMF sampling interval. It shows a 101% figure if any address space is ready to be dispatched but must wait for processor cycles during the interval.

Daily Monitoring for Processor Over commitment

If analysis of the RMF CPU Activity reports show consistently high CPU utilization during a period of poor response times, examine the elapsed and processor times for transactions by IMS message region. If the elapsed time/processor ratio is significantly higher in the lower-priority regions, raise their dispatching priorities relative to other non-IMS work. If all regions are equally affected, see “Minimizing Path Length” on page 281 for information on reducing path lengths. If appropriate, adjust priorities of competing work to move it below IMS message regions.

To monitor utilization by subsystem for processor over commitment, use the RMF Monitor II Address Space State Data report. This report gives processor units consumed by each address space for each RMF measurement interval. IMS control and message regions can be identified by job name.

Detailed Monitoring

Using the IMS Monitor Region Summary report, you can make an initial assessment of dispatching priority problems by examining the Scheduling or DL/I NOT-IWAIT times, that is, the elapsed time not accounted for by IWAIT time. Any delays caused by paging or dispatching for a higher-priority task result in an increase in the NOT-IWAIT times. The SVC Mapping Summary can assist in determining where an SVC is being issued.

Related Reading: For more information on reducing path lengths, see “Minimizing Path Length” on page 281.

Tuning to Remove I/O Resource Contention

A performance problem can occasionally be traced to I/O contention. Tuning activities for I/O devices, DASD storage access, channel usage, or control unit contention should not be confined to system-related devices. For IMS, an important factor is the volume of I/O activity, because reduced I/O activity reduces contention. See “I/O Subsystem Configuration” on page 283 and the considerations for database I/O analysis included in “Program Execution Times” on page 293.

Dynamic Monitoring of the I/O Subsystem

Typically, the I/O subsystem is not monitored dynamically, because corrective action normally requires reconfiguration of devices or data sets. This takes some time to plan and implement. However, if RMF is being run, the RMF Channel Activity and Direct Access Activity can be used to check against your objectives for I/O times, channel utilization, and device utilization.

Immediate action might not be possible. However, it might be possible to move data sets or reconfigure devices to correct the problem before the next IMS restart.

Daily Monitoring

Examination of the RMF Direct Access Device Activity and Channel Activity reports, together with knowledge of the I/O configuration and data set placement, might

suggest a possible tuning action along the lines of those described in the earlier sections on data set placement and I/O subsystem reconfiguration.

Detailed Monitoring

If additional data is required, plan to run GTF and obtain Volume Map, Seek Histogram, and System/Job Summary (EXCP-SIO-IO data) reports. These reports can be used to analyze I/O activity in great detail.

Communication Subsystem Contention

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Because of the great variety of transmission devices and types of lines, each suspected problem in the communication subsystem area must be separately examined. As described in “Considerations for the Communications Network” on page 281, many possible causes of delay exist within the network hardware and software. A variety of different tools might be required to investigate any suspected problems in this area.

Dynamic Monitoring

It is not possible to dynamically monitor the performance of the telecommunication subsystem.

Daily Monitoring

If the response time breakdown data indicates large and variable IMS output queue times, a bottleneck might exist in the IMS output message handling or on a line. Because it is not normally possible to reconfigure the TP network without considerable preparation time, plan to use the IMS Monitor and IMS line traces to investigate those IMS lines.

Detailed Monitoring

VTAMPARS can be used to analyze the GTF SMS records and to tune the VTAM buffer sizes. Sorting of DFSILTA0 and IMSPARS Transit Log data by line or PTERM can isolate line-related problems for output queues to a single line or group of lines.

The IMS Monitor indicates line-related IWAITs, which require tuning of the IMS High Input/Output Pool (HIOP). Although HIOP is dynamically allocated, you might need to adjust either the buffer size definitions or the upper expansion limit. IMS Monitor output that is of interest for contention in communications are:

- The Communication Summary report gives the mean elapsed time for transactions using a VTAM node. This time can be affected by activity in the Message Format pool.

  The report also gives mean NON-IWAIT times. These can reflect device bottlenecks or problems in obtaining an input thread as described in “Save Area Sets” on page 274. The count of times that threads were not available is given in a separate report under the heading “REPORTS”. The line ‘Total Times ECBS waited for SAPS = nnn’ gives the total. This figure is high if the monitor interval includes the time during which network initialization took place.
- The Line Functions report shows, for each active node, the data transmission activity, as described in “Chapter 7. Monitoring Your System” on page 247. The TRANSMISSION BLKSIZE values show the number of characters passed by IMS to VTAM in OUTBUF buffers, a possible indicator of overloading of VTAM buffers. The TURNAROUND intervals can be used to identify inactive lines that might be eligible to distribute transaction entry.

- The Communication IWAIT report gives details of IWAITs for each node.

To trace VTAM activity, you can use GTF (including the Buffer Trace) to examine the content of VTAM buffers as data enters and leaves VTAM buffers. You can also use the Storage Management trace to reflect the overall use VTAM buffers.

Ultimately, tuning of network problems probably requires one or more of the following actions:

- Reduction of volumes of data transmitted. (See “Message Format Options” on page 270.)
- Adjustment of VTAM or NCP options and parameters.
- Reconfiguration of the network to eliminate bottlenecks on multi-dropped lines.
- Isolation and correction of hardware delays caused by device errors.

IMS Message Processing

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

This section identifies possible performance problems associated with message processing and the message queues on an IMS system.

Dynamic Monitoring

MFS and Queue Pool statistics can indicate problems in this area. Pool sizes cannot be dynamically increased. They must be increased at the next IMS start. Normally, once this aspect of performance is tuned, it requires only periodic monitoring.

Daily Monitoring

If IMS output message processing is a problem area, IMS input message processing is probably affected by the same delays. Investigate any large or variable output queue time of IMS message queue and MFS (Format Pool) statistics using the IMSPARS Internal Resource Usage and Message Queue Utilization reports.

If output queue times are high in a DB/DC environment, check the IMSPARS DC Queue Manager Trace report for output message lengths and run the IMS Monitor to study the Communications IWAIT report.

Detailed Monitoring

For detailed monitoring, use the following:

Message Queue Processing

IMSPARS can be used to investigate the IMS Queue Manager in detail, using the DC Queue Manager Trace report. DFSUTR20 and IMSASAP give statistics on frequency and duration of communications-related IWAITs.

GTFPARS Job Summary and Detail Trace reports for the IMS control region can be used to examine the performance of MFS and Message Queue I/O.

Message Queue Data Set Tuning

The three data sets, IMS.QBLKS, IMS.SHMSG, and IMS.LGMSG, contain the directory and the short and long message segments for the message queues. In a busy online system with some constraints on the terminal I/O and conventional SPA pools, these data sets are the most active and critical. Place these data sets on different devices to minimize seek time.

Check to see if the I/O activity is balanced between short messages and long messages. You might have to allow for a possible difference in device access times.

You can do this by examining the IMS Monitor Communication IWAIT report. The entries are organized by line number and reveal high frequency or high I/O elapsed times. Each line in the report identifies the ddname causing the IWAIT, so that you can pick out those that pertain to IMS.QBLKS, IMS.LGMSG, and IMS.SHMSG. The Region IWAIT report similarly identifies message queue I/O and totals IWAIT times for the region.

You can define up to 10 long and 10 short message queues.

Related Reading: For more information on this option, see "Chapter 3. Defining Your System" on page 43 or *IMS/ESA Customization Guide*.

MFS Usage

The GTFPARS Job Summary report for the IMS control region provides data on EXCP-SIO-IO timings to FORMATA/B. The IMS Monitor indicates in the Communication IWAIT report any IWAITs for Format Pool Space or for Directory or Block loading.

Message Format Library Optimization

The active IMS.FORMATA/B data set is the source of all MFS format blocks used during online execution and is usually a busy data set.

You can check the frequency or high I/O elapsed times against this data set using the IMS Monitor Communication IWAIT report. Report lines identified by I/O=DIR=fn or I/O=BLK=fn, where fn is the format block name, show I/O IWAIT time for directory lookup and block fetch by increasing the size of the format buffers. This pool size is specified with the FORMAT keyword on the BUFPOOLS macro for system definition. You can override the value for the online execution using the FBP parameter in the EXEC statement.

To reduce I/O to the directory, you can also create an index of track addresses of the blocks (or only those frequently referenced) and keep this index present in the MFS pool. The index is built when you execute the MFSRVC procedure, specifying INDEX on the utility control statement. You also include the names of the format or message blocks to be in the index.

Multiple reads of the active library data set occur when the size of the MFS control blocks stored on DASD is greater than either the track size or the MVS block size.

Input Queuing and Scheduling/Termination

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

Another set of performance problems is brought about by failure to achieve the performance objectives. The problem might be observed by input queue buildup caused by either increased workload or application program scheduling delays.

Dynamic Monitoring

This section explains how to obtain dynamic monitoring information for IMS input queuing, IMS region occupancy, IMS scheduling, and sync point processing.

IMS Input Queuing: Input queue time is not available. However, using the IMS `/DISPLAY` command with the QUEUE parameter, you can monitor the sizes of the input queues. You can also use the ACTIVE parameter to track performance and detect bottlenecks in scheduling and in the region occupancy area.

IMS Region Occupancy: Not available in any dynamic report. The IMS command `/DISPLAY ACTIVE` is used to monitor region usage.

IMS Scheduling: The PSB and DMB Pools should already be tuned, as recommended in "Initializing SCP and IMS Parameters" on page 268. If all regions are busy, start any necessary additional regions. If some regions are idle, modify classes or PARLIM if one transaction is overloading a region.

Related Reading: For more information, see "Avoiding Contention for IMS Resources (Excluding Buffer Pools)" on page 271.

Sync Point Processing: The occurrences of problems caused by waiting for synchronization points cannot be isolated dynamically.

Daily Monitoring

This section explains how to obtain daily monitoring information for transaction response times, response time breakdown, region occupancy, program scheduling, and sync point processing at program termination.

Transaction Response Times: Extract total (internal) response times for selected transactions that can be used as indicators of system performance.

Related Reading: For more information on sources of response time data, see "Establishing Performance Objectives" on page 248.

Response Time Breakdown: For similar transactions, extract response time breakdown (input queue, switch queue, processing, output queue) from IMSPARS or DFSILTA0.

Region Occupancy: You can obtain indications of the relative use of the dependent regions in a DB/DC environment by using the IMSPARS Availability Reports. Also, you can directly use the summary of region occupancy percentages given in the IMS Monitor Region Summary report.

Program Scheduling: If the response-time breakdown data indicates large and variable input queue times, check the Region Occupancy figures. If all message regions have high occupancy, then another message region might be required. Alternatively, it might be possible to reduce occupancy by reducing program load or program execution times (see below). If some or all of the IMS message regions are not busy, analysis of IMSPARS Transit Time reports by transaction and class probably show that one transaction or class is more critically hit than others. In this case, you should review the designation of classes and the allocation of classes to regions. PROCLIM and PARLIM should be reviewed also. Refer to the earlier information on scheduling options in the IMS options section.

Programs executing as wait-for-input never show 100% occupancy even when they are in the region 100% of the time. Zero occupancy might be cause to review operator procedures, with instructions to manage the number of message regions based upon display of the queues.

The IMSPARS Transit Time Analysis Graphic Summary is useful to analyze input queuing time by time of input across the entire measurement period. This can be

used to discover if high input queue times result from a transient peak in transaction volumes or from a more sustained phenomenon. DFSILTA0 can be used for the same purpose, although its output is numeric rather than graphic.

Sync Point Processing at Program Termination: If you are using IMSPARS to provide response-time breakdown data, a high program switch queue time can indicate that delays are occurring because of a wait-for-synchronization point completion. This occurs if MODE=MULT is used. Check the IMSPARS CPU Usage report for multiple transactions of that type being dequeued per program schedule. As a general rule, use MODE=SNGL to avoid any delays of this type.

Detailed Monitoring

You should examine detailed monitoring information for scheduling and sync point processing.

Scheduling: The IMS Monitor generates reports on PSB and DMB I/O counts and elapsed times. GTFPARS Job Summary and Detail Trace reports for the IMS control region can also be used for detailed investigation of these I/O events. However, scheduling problems are more often related to region availability and message class, PROCLIM, and region class assignments. These are described in “Avoiding Contention for IMS Resources (Excluding Buffer Pools)” on page 271.

Sync Point Processing: Delays caused by waiting for a synchronization point do not normally require detailed investigation.

Program Load and Initialization

This section describes dynamic and daily monitoring and detailed monitoring for program load and initialization.

Dynamic and Daily Monitoring

The resource used to load or initialize a program cannot be isolated or corrected dynamically from normal daily monitoring data. Usually, this aspect of performance, once it is tuned, does not require more than periodic monitoring.

The IMS.PGMLIB data set is specified in the dependent region JCL and contains the application programs. The schedule-to-first DL/I call elapsed time shown on the IMS Monitor Region Summary and Program Summary reports includes the I/O time to bring a program into the region. Included in the I/O time is the system search through JOBLIB (searched first) or STEPLIB. If JOBLIB and STEPLIB are not used for the program library, IMS.PGMLIB should be first in the LNKLSTnn member of SYS1.PARMLIB. Program libraries should always be blocked to full track size. Placement in the dependent region STEPLIB is recommended because a library can be built containing only those members required by the region.

In environments that are not constrained by real storage availability, consider making a trade-off between program loading and preload. You can use the schedule-to-first DL/I call elapsed times from the IMS Monitor Region Summary report to assess the potential savings of preload.

MVS Library Lookaside (LLA) is the strategic method for managing dynamically loaded program libraries.

Related Reading: For more information on LLA, see “Using Library Lookaside under MVS” on page 280.

Detailed Monitoring

The GTFPARS Job Summary and Detail Trace reports for the IMS Message Regions can provide comprehensive data on the counts and sequences of SVC and I/O operations issued to locate and load the user's application program, and to execute the high-level language initialization modules. You can also use these reports to examine dependent region paging I/O if the program load is being managed by Virtual Fetch.

For information about program library options and initialization considerations, see "Choosing IMS Options for Performance" on page 269.

Program Execution Times

You can obtain program execution time information with dynamic monitoring of IMS internal response times, daily monitoring of program execution, and detailed monitoring.

Dynamic Monitoring of IMS Internal Response Times

IMS internal response times are not available in any dynamic report. However, many installations have written special transactions that always perform a fixed (usually small) number of DL/I calls, such as a GHU, ISRT, REPL, DLET sequence. By use of this transaction, an IMS MTO can dynamically monitor the responsiveness of the IMS system. This is not a precise measure, but an indication of what a user is experiencing at that time.

Daily Monitoring of Program Execution

If the response-time breakdown data available from IMSPARS or DFSILTA0 indicates large execution times, the cause might be related to one of the following:

- Program Load Time

  When daily monitoring indicates increases in execution times, the problem might be in the I/O subsystem, as described in "Program Load and Initialization" on page 292.

- Program Initialization and Application Code

  After tuning, execution times are not likely to change significantly.

- DL/I Calls and I/O

  The IMSPARS Internal Resource Usage report gives database I/O statistics. If these are high, the GTFPARS Data Set Overview report can be reviewed, together with the sizes and number of database buffers. Reorganize volatile databases regularly to reduce OSAM or VSAM overflow I/O. For further investigation of this area, the IMS Monitor should be run to obtain Call Summary and Region IWAIT reports. See "Detailed Monitoring" on page 294.

- Other IMS Resource Delays

  The IMS Internal Resource Usage reports in IMSPARS indicate if contention within program isolation activity is high, in which case, plans should be made to run and analyze the IMS PI trace.

- Dependent Region Modifications

  Examine the reasons for high elapsed time for dependent regions or for those application programs that exhibit excessive elapsed time for each transaction. You can use the IMS Monitor Program Summary report. Assuming that the dispatching priority is adequate, you can look for high schedule-to-first DL/I call elapsed times. Setting aside factors related to program load and paging, you need to find out if any modifications have been made in the dependent region code by your installation.

If changes have been made to support accessing MVS data sets or the use of other MVS services, you must evaluate the appropriateness and efficiency of these changes. One reason for these changes might be excessive overlay processing or initialization.

Detailed Monitoring

GTFPARS Job Summary and Detail Trace reports for the IMS Message Regions are the best method of examining the system-related activities of the application program in detail. IMSASAP Program Trace reports provide detailed analysis of application activity related to database access in a DB/DC environment. IMSPARS Database Trace reports show breakdowns of database update activity in a DB/DC environment. DFSPIRP0 should be used to investigate any long delays if PI lockout is suspected as a possible cause.

The time spent in database retrieval that incurs I/O can be a large contributor to program execution time. The frequencies of each type of DL/I call to each database segment type and the frequencies of IMS IWAITS are given in the Call Summary report. If database-related delays are suspected as the cause of high program-execution times, that report helps identify the particular call and database combination that is responsible. The I/O wait times are shown in the IMS Monitor Region IWAIT report.

IWAIT times greater than 100 milliseconds are usually caused by interference from shared DASD or over commitment of resources. Databases using VSAM often experience shorter IWAITs, because the I/O buffer control managed by VSAM is able to meet the request without physical I/O. Excessive chaining of records to overflow tracks can be a contributing factor. Sometimes the data transfer rate of the storage device itself has a limiting effect.

In addition to database design considerations, the two principal strategies for decreasing I/O IWAIT time are:

- Reduce the frequency of I/Os by adding more buffers. This entails identifying inadequate IMS buffer allocation.
- Examine the I/O resource usage to discover if the distribution of data sets is a problem. This is a more protracted analysis that uses detailed output obtained from GTF and RMF traces.

A more extensive action is to change the DL/I structure or database organization.

Related Reading: For more information on changing the DL/I structure or changing database organization, see *IMS/ESA Administration Guide: Database Manager*.

Chapter 9. Modifying Your System Design

This chapter summarizes the planning activities needed when changes are necessary in the IMS system design. The need for changes can result from:

- The integration of new applications
- The staged implementation of an application package
- Modifications to an application design already in production
- Modifications to the network if ETO is not used
- Maintenance of the IMS system at the current level and incorporation of fixes for application program problems
- Tuning of system control parameters
- Redistribution of database data sets
- Storage device changes

All of these changes can affect the IMS system definition and necessitate building a new nucleus and control blocks. The last two changes might only require JCL changes and database reorganization. However, the repercussions can be widespread. With any design change, it is important to control its implementation in order to protect the existing system and the needs of all end users. Implementation of most changes to a production system is best handled by prior testing and a carefully monitored validation period.

The task of administering a change to your IMS online system is likely to involve the following activities:

- Reallocating IMS system data sets and updating execution libraries
- Performing appropriate system definition
- Reviewing security arrangements and reprocessing Security Maintenance utility input
- Documenting changes in operating procedures
- Documenting changes in recovery procedures
- Being aware of the testing strategy and the timing of cutover into production mode
- Analyzing the performance impact and revising predictions
- Coordinating the use of the IMS Monitor and otherwise monitoring the system before and after implementation
- Collecting comparative performance data

In this Chapter:

Assessing Application Changes

Because the design and tuning of an existing production IMS online system involves many people and much machine time, you need to minimize the impact of a change and identify as many potential ramifications as possible.

Application Changes

The checklist shown in the right-hand column of Table 35 summarizes how a change in an application program (or your control of the system) can be assessed for impact. It shows the IMS area affected and suggests what other administrative areas of responsibility might be affected.

If you are notified of an application program design change or are tracking a staged implementation, you must extract from the documentation the necessary physical system definition changes. You also need suitable detail to enable you to assess whether your defined environment and operational control are affected.

The first column of Table 35, “Change Event”, shows the kind of change you isolate from the documentation. For example, an application system adds a batch message program to generate a report after closedown of a physical part of the network. The following information can be obtained:

Programs Name and PSB, program size, and use of overlay

Databases Already defined, read-only access

Transactions None

Message formats
None

Output Name of spooled output procedure

Terminals None

Network Control
Whether any change in network availability

Scheduling BMP region: time, frequency, limits

Exit routines None

Tuning N/A

Security No AGN control

Working with this information, you define the APPLCTN macro, perform system definition (CTLBLKS), and tailor the BMP region JCL. Then you add its BMP scheduling instructions to the operations procedure, including a spooled output print step. An alternative strategy is to perform a MODBLKS system definition and use online change for the cutover to production mode.

Table 35. Checklist for Application Change Control

Change Event	Area Affected	Impact/Action
Programs modified		
Coding fix	IMS.PGMLIB	Documentation, log the change
Database access	IMS.PGMLIB IMS.PSBLIB IMS.ACBLIB	Monitoring DL/I calls Use of online change
Addition	Above plus APPLCTN macro	Pools and resource contention Use of online change
Databases		
Access	IMS.ACBLIB	Monitoring
Structure	IMS.ACBLIB	Database reorganization

Table 35. Checklist for Application Change Control (continued)

Change Event	Area Affected	Impact/Action
Additions	JCL, IMS.ACBLIB DATABASE macro	Operations and recovery procedures Use of online change
Transactions		
Change content	TRANSACT macro	Recalculate message queues, monitoring
Added	TRANSACT macro	Scheduling algorithm, use of online change
Workload	Message Queues	Recalculate blocking
Message formats	IMS.FORMAT	Buffers and monitoring Use of online change
Output		
MVS file	JCL for devices	Operation procedures
Spool print	LINEGRP, LINE macros	Operation procedures
LTERM	TERMINAL macro	Alternative destinations, operational changes
Terminals		
VTAM	System definition macros	VTAM generation, operating procedures
ETO	IMS.PROCLIB	ETO generation
Other	System definition macros	JCL for online, operating procedures
Network control	System definition macros	VTAM initialization, MTO procedures
Scheduling	TRANSACT macro	Message class algorithms operating procedures
Exit routines		
DL/I	IMS.RESLIB	Monitor DL/I calls
Message edit	IMS.USERLIB	Ensure against abnormal termination
Security	IMS.USERLIB	Ensure security needs are met
ETO	IMS.RESLIB	Monitor ETO
User descriptors	IMS.PROCLIB	
Tuning		
JCL parameters	IMS.PROCLIB	Coordinate operators change
System definition	IMSCTF macro	Monitoring
Security	SECURITY macro	Coordinate operations
Signon	Security Maintenance utility, exit routines, and RACF	Coordinate Security Maintenance utility (or other security function) with nucleus and validate passwords

Table 35. Checklist for Application Change Control (continued)

Change Event	Area Affected	Impact/Action
Terminal	Security Maintenance utility, exit routines, and RACF	Coordinate Security Maintenance utility (or other security function) with nucleus and validate passwords
Transaction	Security Maintenance utility, exit routines, and RACF	Coordinate Security Maintenance utility (or other security function) with nucleus and validate passwords
AGN	Security Maintenance utility and JCL	Coordinate Security Maintenance utility with nucleus and validate passwords
All of the above	IMS.MATRIX	Coordinate with online change

Planning for System Definition Changes

The IMS online system design reflected in the macro specifications is subject to change. This design is rarely a one-time execution that occurs at initial installation. The allocation of real resources and terminal connections change, if not because of hardware changes, then because of application workload and design changes. The ongoing task of monitoring for performance is likely to modify elements of the online system design.

Controlling System Definition Processing

When planning a change to system definition, you usually have to coordinate a series of macro changes. The IMS system definition stage 1 input is suggested as a point of control. Review changes to the macros for the accuracy of the parameter values and add adequate explanation of how the value was derived. You should have a control document that records the nature of the proposed change, the reason for it, and suitable authorization. You can then complete the record by entering the action taken, or planned, for each item.

You must also coordinate the timing of the system definition stage 2 processing. The new version of the control program, with any required JCL changes, must be tied to an operational cutover.

Re-specifying a parameter on one or more macro statements does not require a full system definition or the processing of a complete stage 2 jobstream. You can make many tuning changes by overriding parameter values with others specified in the JCL statements. You can delay making the new values a permanent part of the control program until a more extensive change occurs in the macro content. Consider that all such overrides need to be coordinated with PROCLIB contents and operator instructions.

Determining the Type of System Definition Required

Your requirement to perform a full system definition or a partial one depends on the macro statements altered. However, the modification of a particular keyword might be the only cause for full generation. Because the amount of processing performed during a stage 2 execution is significant, you should plan to take advantage, wherever possible, of economics in the type of generation selected.

Related Reading: For a summary of types of system definition to choose, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Making Changes to the Network Definition

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

You can frequently change terminals that are connected to an IMS online system having attendant buffer storage tuning. If you are introducing static VTAM terminals or new types of BTAM terminals into the system, you must regenerate the nucleus. (An online system definition is necessary if no VTAM support was At the same time, you can delete support for specific terminal types.

With BTAM terminals, adding a specific terminal type or switched terminal support requires at least a NUCLEUS generation.

If you are introducing VTAM terminals and using ETO, it is not necessary to regenerate the nucleus. When you add an ETO terminal supported by the existing descriptors, IMS uses the appropriate ETO logon descriptor to define the terminal.

Related Reading: For more information on changing ETO logon descriptors and ETO terminals, see *IMS/ESA Administration Guide: Transaction Manager*.

When considering deleting terminals, leaving the definitions in place might be more efficient if the terminal is physically removed and messages cannot be routed to that LTERM. You can make the deletions in a later definition change.

If you are adding MSC for the first time or adding physical links, you need an online system definition. If, because of redesign of the number of IMS systems, you are deleting a requirement for main storage-to-main storage or MSC VTAM connections, you only need a NUCLEUS generation. If you are changing link or system identification names, link-edit the regenerated control blocks.

Coordinating System Definition and Current Security Definitions

When you are altering the specifications that control security options, you must coordinate the execution of system definition and the Security Maintenance utility, as well as changes to operating procedures. If you have used IMSGEN and COMM macros to specify security options, you can use the SECURITY macro to override any of the options.

The keyword SECLVL on the SECURITY macro determines what overrides are available to the MTO during restart. If you change any of the values for this keyword but not those for TYPE, you only need to perform a CTLBLKS generation and link-edit. The operational impact is described in detail in “Security Considerations for the Master Terminal” on page 124.

System Definition Changes

Remember that any change that requires a new or re-link-edited nucleus must be followed by execution of the SECURITY procedure to build into the control program the current security options and matrixes.

Making System Tuning Changes

Through ongoing monitoring and awareness of end-user feedback, you should be aware of potential tuning activities. changes by the degree of impact to the overall design of the online system.

The tools and strategies for performance-related tuning activities are described in “Chapter 7. Monitoring Your System” on page 247, and “Chapter 8. Tuning Your System” on page 265. These chapters assume that performance administration is an iterative task made up of:

Data collection
Using monitors and traces

Data reduction
Using report processors and manual procedures

Data analysis
Finding problem indicators

Change implementation
Proposing solutions and modifying the system

This plan should include a tuning and maintenance section. You can combine items from this section with other required changes to save additional system definition processing and reduce the frequency of changes to the operating procedures. An overall design change log enables you to keep track of the proposals that are analyzed and approved, as well as those under consideration for future implementation. A control document like this also assists in detecting conflicting changes.

The documentation for a change that affects the IMS online system design should include the reasons for the change and a checklist of what parameters or components in the system require modification. Ideally, the documentation should have sections on operations, database maintenance, recovery, security, and monitoring arrangements.

Resource Utilization Changes

Changes that are slanted toward more efficient use of virtual storage and minimizing I/O for an existing production system involve:

- Re-specifying EXEC parameters for the control region
- Allocating DASD storage for message queues
- Database reorganization and data set distribution

Application and Database Design Changes

As a result of a performance study for an application package, changes to the online system design can range from a simple update of IMS.PGMLIB, to database redefinition in IMS.DBDLIB and IMS.ACBLIB, If application programs are changed in IMS.PGMLIB, the message processing regions must be stopped and restarted to store the correct directory entry in the BLDL list.

Communication Design Changes

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X		X

If you detect a bottleneck whose symptom is the transmission rate between a communication device and IMS, the response falls into the same planning as for network definition changes.

Managing Online System Definition Changes

A request to add an application or to modify the current set of programs, transactions, and database usage need not force a complete system definition and an IMS restart, with possible interruption for the online users. You can examine the request to see if an online change can be made.

If the request does not involve changes to the IMS network or the use of static terminals as defined in the current IMS system definition, you can arrange for the changes to be made while the IMS system is executing online. Manage the preparation of the IMS system data sets as described under “Administering IMS System Data Sets for Online Change Function” on page 82. It is also important to assess the impact to the terminal users and the online system operation.

Deciding If System Modifications Can Use Online Change

A checklist of the changes that can be serviced by online change is given in Table 36. The list is based on the effect on the stage 1 system definition macros, because the stage 1 input is regarded as the control point.

Table 36. System Definition Resource Modifications Allowed for Online Change

System Definition Macro	Permissible Online Changes
APPLCTN	Add a PSB (application) and its attributes Change attributes Delete a PSB
DATABASE	Add a database and its attributes Change attributes Delete a database
RTCODE	Add a routing code and inquiry attributes Delete a routing code
TRANSACT	Add a transaction and its attributes Change attributes Delete a transaction

Remember that these changes require a MODBLKS generation, and that the corresponding security changes require a subsequent execution of the Security Maintenance utility. Also, the system definition preprocessor can be a useful part of your preparation. When adding or changing resource names, the preprocessor can detect any invalid names or duplicate names and help ensure a successful system definition run. If the LGEN subparameter is specified in the SYSTEM= keyword on the IMSCTRL source statement, stage 1 processing occurs during execution of the preprocessor. Some limitations to the modifications are explained in the next section. You cannot make modifications that affect the terminal network.

Planning Considerations for Online Change

When assessing whether a set of changes can be handled with an online change, you must take into account several limitations. In general, the checking performed by the stage 1 processing does not tell you if you have made a change that cannot be implemented online. You need to consider effects of the following:

- APPLCTN macro
- DATABASE macro
- RTCODE macro
- TRANSACT macro
- Page fixing
- EMHB size

APPLCTN Macro

If the message class is assigned as part of the PGMTYPE parameter, that class cannot exceed the maximum number of message classes currently defined for the system.

If the transaction is designated for Fast Path, Fast Path must be active in the system.

Routing a transaction to another IMS system requires that the system name (SYSID parameter) and the use of MSC are not newly defined for the MODBLKS generation.

Although you can make changes to the RESIDENT and DOPT characteristics, PSBs defined as RESIDENT operate as nonresident until after the next restart, because the action of making PSBs resident takes place at IMS system initialization time.

Changing the scheduling attribute to a resident PSB causes that PSB to become nonresident until the next IMS restart.

If a BMP program becomes a message processing program, the transaction characteristics defined in the TRANSACT macro that control message scheduling do not take effect until after the next restart. However, the MTO can use the `/ASSIGN` command to specify appropriate message class and processing priorities for the particular transaction. The transaction then becomes eligible for normal message scheduling.

DATABASE Macro

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

Although the RESIDENT characteristic can be added, the process of making DMBs associated with the database resident does not take effect until after the next restart of the IMS online system.

Changes to the ACCESS parameter are not part of online change; this change can be handled with the `/START DATABASE` command.

You cannot include any kind of change to MSDBs.

RTCODE Macro

The addition of this macro statement, or changes to its specification, is only allowed if Fast Path is active in the system. Make sure that the existing Fast Path User Input Edit routine is able to handle any added routing codes.

TRANSACT Macro

The MTO can control several of the characteristics specified by this macro using such commands as /ASSIGN, /MSASSIGN, /START, and /STOP. Any changes you make to the TRANSACT macro characteristics are not implemented as part of the online change processing and only become effective at the next cold start of the IMS online system. They are:

```
PRTY  PROCLIM (count value)  PARLIM
SEGNO  SEGSIZE  SYSID
```

Transactions designated as Fast Path potential need Fast Path to be active in the current system.

Routing a transaction to another IMS system requires that MSC facilities be active in the current system. You cannot introduce a system name (SYSID parameter) that was not previously defined in the current system.

Edit exit routines specified for the transaction must already be part of the current IMS online system.

Page Fixing

No additional page fixing is done for added control blocks until the next restart of IMS.

EMHB Size

If you use online change to add or change a transaction-specific EMHB size, ensure that the new EMHB size is not larger than the EPSESRT size. The EPSESRT size is determined only during initialization.

During normal transaction processing, IMS checks the size of the input message against the EMHB length and the EPSESRT length. If the input message exceeds either the EMHB length or the EPSESRT length, it is rejected with message DFS0444.

Performing Capacity Planning

Previous sections have been concerned with short- or medium-range planning for changes in the processing requirements of your IMS online design. However, you might want to examine the long-term adequacy of your design. For example, if you extrapolate trends in workload, will enough computing power be available?

You can use:

- Individual transaction profiles
- An overall processing profile
- An estimate of percentage processor utilization

As a result of your performance monitoring, you can establish transaction profiles. One part of these profiles can be a trend analysis. You must determine mean and peak arrival rates: currently predicted, sample actuals, and future expectations. The overall processing summary can be taken from the IMS Monitor Run Profile report.

Performing Capacity Planning

One way to investigate limiting capacity is to extrapolate the volume of transactions. You can establish a trend from percentage processor usage for several monitor points. A common observation from performance measurements indicates that the increase in machine cycles is nearly linear with the increase in transactions per second on systems dedicated to IMS. If you plot several points, showing increasing transaction loads against percentage utilization, you can extrapolate to find your limiting number of transactions as the percentage utilization approaches 100%.

You can also see if a predicted maximum workload corresponds to a utilization that is below a practical upper limit. The workload is estimated by weighing the current overall profile with additional transactions.

In making such predictions, you should have adequate virtual storage and an I/O configuration sufficient to handle the increased transaction load. A careful consideration of remaining available I/O and virtual storage resources is necessary, because contention for these elements has a more severe impact on performance than the level of processor utilization.

If you are managing a large system running under MVS, you might want to reduce your requirements for CSA. You can plan to use the DL/I separate address space and, if necessary, local storage for IRLM for lock management. Using your projected workload, you might still want to estimate virtual storage with various configurations.

Chapter 10. Administering a Data Sharing Environment

APPLICABLE ENVIRONMENTS	DB/DC	DBCTL	DCCTL
	X	X	

This chapter describes system administration activities when you allow more than one IMS online system or batch system concurrent access to data that resides in a common database. If you use IMS to control this common access to data, the use of data sharing support provided by Database Recovery Control (DBRC) is required.

In this Chapter:

Data Sharing Concepts and Terminology

An IMS system includes a set of databases that are potentially available to all the declared application programs. Access to an individual database is a characteristic defined in a program's PSB. Data sharing support makes it possible for application programs in separate IMS systems to have concurrent access to databases. To ensure that database changes at the segment level originating from one program are fully committed before other programs can access that segment's data, IMS systems use lock management.

DBRC and Data Sharing Support

Concurrent access to databases by systems in one or more processors is controlled with a common (shared) Database Recovery Control (DBRC) RECON data set. To maintain data integrity, status indicators in the RECON data set control concurrent access and recovery actions for the databases. This common RECON data set is required in a data sharing environment so that a given database has a DMB number that uniquely identifies it to all the sharing subsystems. The DMB number that DBRC records in its RECON data set is related to the order in which databases are registered to DBRC. Using multiple RECON data sets can result in the same DMB number existing in each RECON data set for different databases. This condition can result in damage to databases.

Two Levels of Control

With data sharing, two levels of control are possible:

- You can prevent concurrent database access and scheduling of application programs that might jeopardize database integrity.
- You can use a global block locking scheme to maintain database integrity during concurrent access of a database.

Database as a Data Resource

Some differences exist in support for data sharing configurations. Generally, a complete database is regarded as a data resource. When invoked within an IMS online system, or as a batch IMS system, the data resource must be available for an individual application program to process. The resource is not available if, for example, a data resource is used exclusively by one IMS system or is flagged as needing recovery.

For DEDBs, the data resource is further divided; each individual area is considered a unit of data resource. When this chapter refers to “database”, it is equivalent to a DEDB area, unless otherwise noted.

Data Sharing Restrictions

Some overall restrictions apply:

- Batch IMS system support excludes use of MSDBs and DEDBs.
- Only IMS online systems that utilize Fast Path can share DEDBs.
- Data sharing support excludes MSDBs and GSAM databases.

If you are using the High Speed Sequential Processing (HSSP) function, you cannot share data with previous releases of Fast Path. If you attempt to share data with prior releases, database integrity might not be protected.

DBRC Authorization and Data Sharing Control

Controlling which IMS systems are authorized for concurrent access to a database requires the data sharing support provided by DBRC. IMS systems perform an automatic signon to DBRC, and this action ensures that DBRC knows which IMS systems (and utilities) are currently participating in shared access. Subsequently, a system’s eligibility to be authorized for access to a database depends on the declared degree of sharing permitted and other status indicators in the RECON data set.

DBRC denies access when the database status indicates that recovery is required or backup procedures are in progress.

Databases that are to take part in data sharing must be registered in RECON. Each registered database has a current status that reflects whether it can take part in sharing and the *scope* of the sharing. The concept of scope combines several ideas:

- The type of access—read or update
- Whether more than one access can occur within the database simultaneously
- Whether an IMS system desiring access is in the same or a different processor

How you specify these factors is explained later in this chapter.

Database Integrity Without Data Sharing Support

Without data sharing support, the responsibility for database integrity lies with administration and operations. Two situations where data integrity is not protected are:

- In a single processor, with the database JCL specifying a disposition of SHR, multiple IMS online or batch systems executing simultaneously might concurrently access the database without integrity.
- If multiple IMS systems are executing in more than one processor and a database is on shared DASD, those systems can be concurrently accessing the database. This concurrent access cannot be directly controlled by JCL, even when a disposition of OLD is specified.

To understand data sharing, you need to understand how applications and IMS subsystems share data.

How Applications Share Data (Process Option)

The processing options for an application program are declared in the PSB and express the intent of the program regarding data access and alteration. They are specified with the PROCOPT keyword as part of the group of statements that make up the PCB for a particular database access. The PCB declaration implies a processing intent.

If the application program is to insert, delete, replace, or perform a combination of these actions, the application program is said to have update access. An online program having exclusive access, specified as PROCOPT=E, is interpreted as having *update access*.

Programs that need access to a database but do not update the data can do so in two ways. They can access the data with the assurance that any pending changes have been committed by the program that instigated the change; this is termed *read access*. Alternatively, they can read uncommitted data, if the program does not specify protection of data status. This is termed *read-only access*.

Related Reading: For more information on PROCOPT values, see *IMS/ESA Utilities Reference: System*.

How IMS Subsystems Share Databases (Access Management)

This section explains the concept of *access*, the two levels of data sharing (database and block), and how data access is controlled by each level of sharing.

Establishing Database Access

The ACCESS keyword parameter in the DATABASE macro specifies how the subsystem requesting access to a database plans to use the database. The possible values are *update*, *exclusive*, *read*, or *read-only*.

Related Reading: For a detailed description of the ACCESS keyword, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

If the subsystem requires exclusive use of the database, it can declare its access as *exclusive*. This allows the subsystem to insert, delete, replace, or perform a combination of these actions, but it disallows data sharing in block-level or database-level environments configurations. If the subsystem needs to insert, delete, replace, or perform a combination of these actions and must participate in data sharing, it can declare its access as *update*.

For application programs that do not require update or exclusive access, you must find out what kind of read access is required. Subsystems that plan to read from a

database, but not update the database, can do so in two ways: *read* access and *read-only* access. Read access means that the subsystem can access the data with the assurance that any pending changes have been committed by the program that requested the change. Read-only access means that the subsystem is allowed to read uncommitted data.

How you specify subsystem access depends on whether you are running a batch or online system.

Declaring Access for IMS Batch Systems

For an IMS batch system, the access (for a particular database) directly corresponds to the highest PROCOPT value specified in the program's PCBs or SENSEG statements. For example, if one PCB has PROCOPT set to G and another PCB for the same database has PROCOPT set to I, the value of I is used; that is, the access for that database is update.

Declaring and Changing Access for Online Systems

For an online IMS system, you specify access with the ACCESS keyword (in the DATABASE macro) during system definition. The access is declared for each individual database and reflects a desired access suitable for all the application programs that might be scheduled against that database.

Related Reading: For a detailed description of the ACCESS keyword, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

You can dynamically change the access of an online IMS system for an individual database with the `/START` command.

Related Reading: For a description of the `/START` command, see *IMS/ESA Operator's Reference*.

Levels of Sharing

IMS provides two kinds of access management:

- Sharing at the database level
- Sharing at the block level

Sharing at the Database Level

The complete database or DEDB area is a resource that can be accessed for update by only one IMS system at a time. For area resources, this is termed *area-level sharing*.

Data access is protected so that:

- One IMS system is authorized to update the database, and other authorized IMS systems can have read-only access.
- Multiple authorized IMS systems can concurrently schedule the database with read or read-only access.

An IMS system in the data-sharing environment can be online or batch. For area-level sharing, participating IMS systems must be online. An IMS utility can be thought of as a batch system that is able to function with database-level sharing.

Sharing at the Block Level

Within a single processor, or across communicating processors, multiple IMS online or batch systems can update a database concurrently. Data integrity is preserved

for the multiple IMS online or batch systems that concurrently access the shared data. *Block-level sharing* for DEDBs is between IMS online systems only.

Sequential dependent segments are supported.

Within a database, resources are reserved at the block level. For OSAM databases, the block is a physical block stored on a direct access storage device. For VSAM databases and DEDBs, the block is a control interval (CI).

Related Reading: For more information on OSAM databases, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

When the IMS subsystems participating in block-level sharing are restricted to one MVS system, this type of sharing is termed *intra-MVS sharing*. When the IMS subsystems participating in block-level sharing are operating in separate MVS systems, this type of sharing is termed *inter-MVS sharing*.

Block-Level Sharing of VSO DEDB Areas: Multiple IMS subsystems can concurrently read and update VSO DEDB data. The three main participants in this process are:

- The coupling facility hardware
- The coupling facility policy software
- The XES and MVS services

The *coupling facility hardware* provides high-performance, random-access shared storage in which IMS subsystems can share data in a sysplex environment. The shared storage area in the coupling facility is divided into sections called *structures*. For VSO DEDB data, the structure type used is a *cache structure* (as opposed to a list structure or a lock structure). The cache structure is designed for high-performance read reference reuse and deferred write of modified data. The coupling facility and structures are defined in a common MVS data set, the *couple data set* (COUPLExx).

The *coupling facility policy software* and its cache structure services provide interfaces and services to MVS that allow sharing of VSO DEDB data in shared storage. Shared storage controls VSO DEDB reads and writes, as follows:

- A read of a VSO CI brings the CI into the coupling facility from DASD.
- A write of an updated VSO CI copies the CI to the coupling facility from main storage, and marks it as changed.
- Changed CI data is periodically written back to DASD.

The *XES and MVS services* provide a way of manipulating the data within the cache structures. They provide high performance, data integrity, and data coherency for multiple IMS subsystems sharing data.

The Coupling Facility and Shared Storage: Each VSO DEDB area is represented in the coupling facility shared storage by one cache structure. These cache structures are **nonpersistent**. That is, they are deleted after the last IMS subsystem disconnects from the coupling facility.

In the following description, one VSO DEDB area holds data that two or more IMS subsystems in a sysplex are to share. Figure 43 on page 310 represents the coupling facility shared storage that provides structure storage of varying sizes (S1

through S9). The cache structure is to be used for VSO DEDB data. The structure name `A1$0XXX111222333` represents the VSO area.

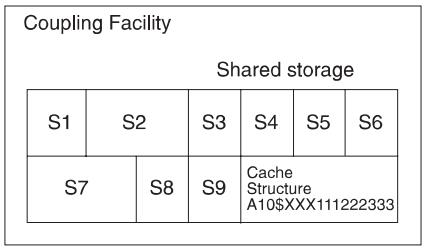


Figure 43. Coupling Facility Shared Storage

Each cache structure in the coupling facility is composed of a directory portion and a data portion.

The directory portion is the basic unit of interaction with the coupling facility. It contains a name that consists of the characters VSO, the area name, and the relative byte address (RBA); for example, `vsoarea1rba1`. Only one directory can exist in a particular CI, so each CI within the area and within the cache structure is uniquely identified.

The detail of `CACHE STRUCTURE A10$XXX111222333` is represented in Figure 44.

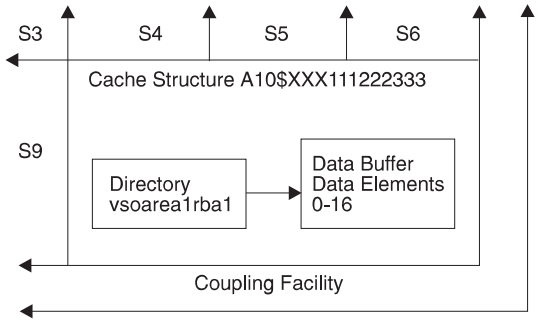


Figure 44. Detailed View of CACHE STRUCTURE A10$XXX111222333

Duplexing Structures: Duplexing structures are duplicate structures for the same area.

Private Buffer Pools: IMS provides special private buffer pools for Shared VSO areas. Each pool can be associated with an area, a DBD, or a specific group of areas. These pools are only used for Shared VSO data. You can use the private buffer pools to request buffer lookaside for the data. The keyword LKASID or NOLKASID, when specified on the DBRC commands INIT.DBDS or CHANGE.DBDS, indicates whether or not to use this lookaside capability.

Controlling Data Access

DBRC is an integral part of the execution of an IMS batch or online system when data sharing control is active. IMS support differs for database-level sharing and block-level sharing.

Database-Level Sharing

When two or more IMS systems are executing concurrently and sharing data at the database level, they can be in the same or different processors. Figure 45 shows the configuration of a batch IMS system and an online IMS system in one processor, and a second online IMS system in another processor. (Fast Path application programs might be executing within the IMS online systems.) Each system must obtain authorization from DBRC before it can access the database, even if the database is not registered for sharing.

Figure 45 shows a database as a shared resource and also illustrates area-level sharing. One or more areas making up the DEDB could be a shared resource. The RECON data set is also shared as a VSAM key-sequenced data set (KSDS), with status being read and updated by a DBRC control portion of the online or batch IMS system. For IMS online systems, DBRC executes in a separate region, automatically started by the control region at initialization. For a batch IMS system, DBRC control is contained within the batch region, as it is for database-level sharing.

The RECON data set keeps track of the:

- Sharing level allowed for each database
- Databases or areas currently authorized for processing
- IMS systems that are involved
- Status of all of those systems
- Database status from a recovery viewpoint

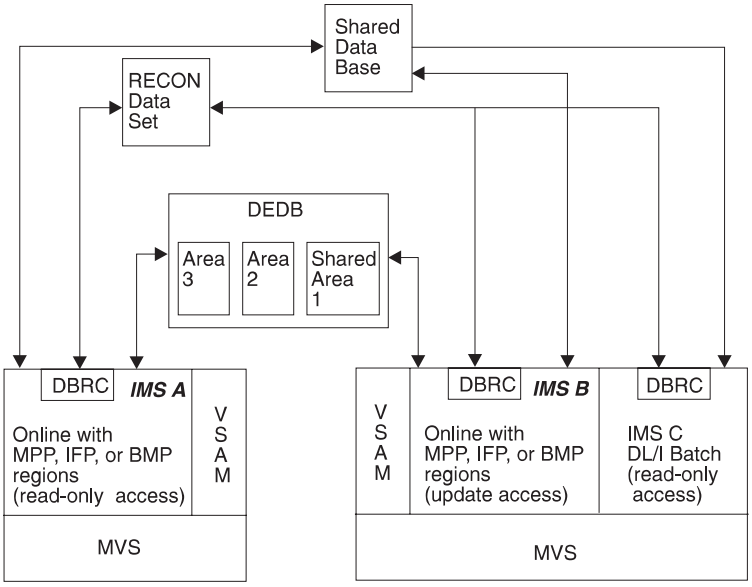


Figure 45. Database-Level Sharing

Using IRLM with Database-Level Sharing

When IRLM is used by IMS for locking services, it provides **all** locking services.

When all of the IMS batch and online instances use one or more communicating IRLMs with database-level sharing, database extensions and buffer invalidations are notified to the read-only IMS instances. This is the same as full block-level data sharing, but without the locking activity.

With the assistance of IRLM in accomplishing database-level sharing, programs that read without integrity see valid, but perhaps uncommitted, data more frequently. This does not provide any guarantees that read without integrity problems are solved (see *IMS/ESA Application Programming: Design Guide*), but it does improve database-level sharing. Because of the order in which all changed blocks or control intervals (CIs) are written to DASD, including **when** they are written, a program reading without integrity can still experience inconsistencies from one execution to another.

IMS Database-Level Sharing with IRLM: IMS allows use of the coupling facility for buffer coherency works for both database-level sharing and block-level data sharing.

The level of sharing on an individual IMS database is controlled by the SHARELVL specification in the DBRC registration for that individual database.

Some databases can be shared at the database level, others are shared at the block level, and some are not shared at all. The type of sharing used for a database should be based on the data integrity and availability needs of all the application programs using it.

Block-Level Sharing

The block-level sharing environment is characterized by the presence of an independent component, the Internal Resource Lock Manager (IRLM). If multiple processors are operating under MVS and each contains an IMS online system participating in block-level sharing, an IRLM component must exist in each operating system. The IRLM manages all requests for locks that control access to resources participating in data sharing.

Each IRLM responds to the requests that originate from its associated IMS systems. To do this, it maintains records of the participating subsystems, as well as the status of locks that are held and waiting.

When the IRLM is used by an IMS online system, it also provides the lock management that controls database resources that are concurrently accessed by application programs within the online system. In this way, the IRLM associated with an IMS online system provides almost all locking services for that system.

The IRLMs communicate with each other and manage the status of locks held for different database locks.

When using the IRLM, you must define DBRC-registered SHARELVL=1 VSAM databases to VSAM with the appropriate SHAREOPTIONS required for block-level data sharing. This might require you to make a VSAM definitional change.

Figure 45 on page 311 illustrates an environment where two processors containing online IMS systems are sharing data. It shows a database as a shared resource and illustrates DEDB area sharing. The IRLM region executes separately as a system task started from the system console.

Also, the figure shows that the RECON data set is shared with status being read and updated by DBRC.

Testing Inter-MVS Sharing: Multiple IRLMs can take part in the control of data sharing at the block level when IMS subsystems are executing in the same

processor. This could be the case when IMS online systems are being tested for their use of data sharing before one of the systems is installed on a second processor.

A special case of inter-MVS sharing occurs when multiple IRLMs execute on a single processor. In this case, you declare global sharing and give each IRLM a unique MVS subsystem name and an identifying IRLM number (ID).

Examples of Data Sharing Configurations

Following are three examples of data sharing, at the:

- Database level with update activity
- Database level with multiple readers
- Block level

Data Sharing at the Database Level with Update Activity

This type of sharing allows one IMS batch or online system to have update access. All other IMS systems must have read-only access. For full-function databases, uncommitted data can be read. For Fast Path, uncommitted data does not exist, but all of the updates for a transaction might not yet have been written. This can cause invalid pointers to be referenced when Get Next processing is performed.

Normal program isolation protects the integrity of the updating system, but you can use the IRLM for lock management on an IMS online system. Figure 46 illustrates an online system with update access sharing data with two batch systems that are authorized for read-only access to the same data.

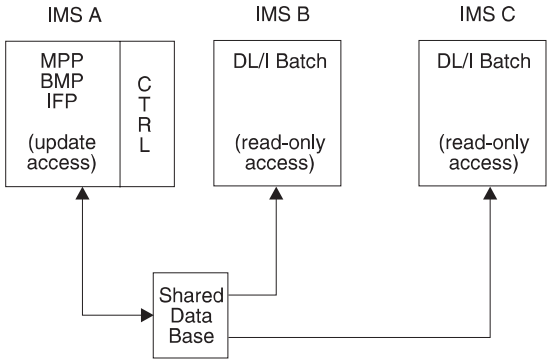


Figure 46. Example of Data Sharing at the Database Level with Update Access

Data Sharing at the Database Level with Multiple Readers

This type of sharing allows multiple authorized IMS subsystems to read data concurrently with read or read-only access. A sample configuration is shown in Figure 47 on page 314. Data integrity is not compromised, because no updates are allowed in this configuration.

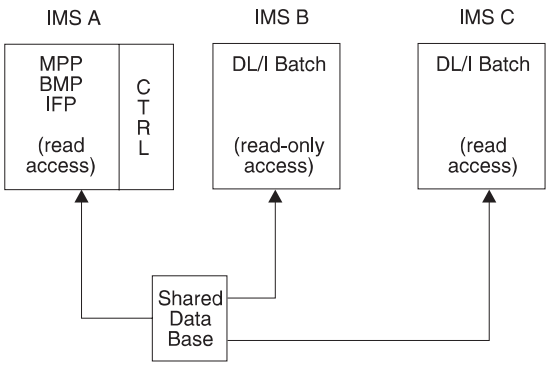


Figure 47. Example of Data Sharing at the Database Level with Read Access

Data Sharing at the Block Level

Online IMS systems with full update capability can share data among themselves and can allow other batch IMS systems to read committed data or update the data. The batch system can have the data integrity protected or use read-only access. This type of configuration is illustrated in Figure 48.

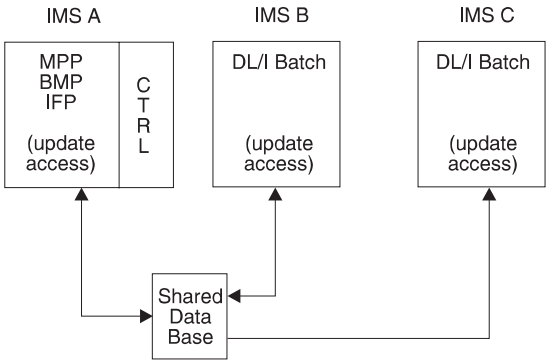


Figure 48. Example of Data Sharing at the Block Level with Update Access

Data Sharing Administration Activities

When an online IMS system must be available for long periods, the opportunity to schedule batch systems to process against some or all of the databases becomes difficult. Applications such as report generators need controlled access to common data.

In an installation with more than one operational online system, the control of databases might need to be shared. This is the case for distributed processing or when a major application system needs data that is maintained by an application system operating on another processor.

In this section, two administration activities are described:

- Assigning a sharing level with DBRC
- Establishing naming conventions

Assigning a Sharing Level with DBRC

By automatically updating the status information in the RECON data set, DBRC records all recovery-related information for a chosen set of database data sets.

DBRC also records system log information, successful reorganizations, database image copies, and completed database backouts. If you use the Log Recovery utility, changes for the system log data sets are also recorded.

Through the report capabilities of DBRC that list the content of the RECON data set, you can trace the physical input that participates in recovery: the system log data sets, image copies, and change accumulations. When recovery utilities are executed, DBRC checks the input data sets for accuracy against the RECON data set status.

GSAM and MSDB databases cannot be shared and should not be registered to DBRC.

Related Reading: For more information on assigning a sharing level with DBRC, see *IMS/ESA DBRC Guide and Reference*.

Establishing Naming Conventions

If you decide to use data sharing and have additional databases to control with DBRC, you need to review the data set naming conventions established for database data sets.

You might consider indicating, using the chosen resource names, that a database participates in data sharing. A program that accesses a shared database might also have the PSB name indicate a read-only property.

You and a database administrator should review naming conventions for the following resources:

- Databases, their ddnames and data set names
- Image copy and change accumulation data set names
- Online log data sets (OLDSs)
- System log data sets (SLDSs)
- PSB and program names
- Transaction codes
- Subsystem identifying names for IMS systems and IRLMs

Tailoring IMS Systems That Share Data

You tailor an IMS system to meet your data sharing requirements by:

- Using system definition macros to:
  - Include DBRC data sharing support for IMS batch systems
  - Include the IRLM (for block-level sharing)
  - Declare database access attributes

  See “Including DBRC and the IRLM” on page 316 and “Declaring Online Databases That Share Data” on page 317.
- Initializing DBRC data sets and JCL. See “Initializing DBRC Support” on page 318.
- Tailoring the execution JCL and virtual storage usage. See “Tailoring Execution JCL” on page 318.

Data Sharing System Definition

Within the complete installation process, several activities must be completed before executing the IMS subsystems that use database sharing. Table 37 lists the installation activities, indicates whether an activity affects database sharing, and identifies additional activity required.

Table 37. Installation Steps with Additional Data Sharing Activity

Step	Installation Activity	No Sharing	Sharing	Additional Activity
1	Build system libraries—IRLM	No	Yes	Add IRLM to system library.
2	Allocate and catalog IMS system data sets	Yes	Yes	Prepare RECON data set. Initialize DBRC controls. Coordinate DL/I exits.
3	Prepare system definition and JCL	Yes	Yes	Macros and JCL changes. Perform system definition.
4	Tailor the operating system	No	Yes	Add VTAM communication. Add IRLM subsystem.
5	Tailor IMS performance options	Yes	Yes	Define buffers.
6	Build DBDLIB	No	No	(No change)
7	Build PSBLIB	Yes	Yes	Review PROCOPT values.
8	Build ACBLIB.	Yes	Yes	Perform ACBGEN
9	Selections for dynamic allocation	Yes	Yes	Review options.
10	Prepare MFS libraries	No	No	(No change)
11	Prepare program libraries	Yes	Yes	Review call sequences.
12	Perform initial database loading	No	No	(No change)
13	Establish security	No	No	(No change)
14	Initialize IMS.MODSTAT	Yes	Yes	Coordinate ACBLIB use and DBDLIB and PSBLIB.
15	Copy staging libraries to active libraries	Yes	Yes	Coordinate ACBLIB use.

Including DBRC and the IRLM

The procedure for defining systems for data sharing depends on whether you are running an online or batch system.

IMS Online Systems

To control data sharing, the data must be registered to DBRC. For block-level sharing, the IRLM must be active in your system.

If the IRLMNM parameter is not specified (in the IMSCTRL macro) and you want to activate the IRLM, you must code IRLM=Y in the execution JCL. When the

IRLMNM parameter is not specified in the IMSCTRL macro or in the execution JCL, and IRLM=Y is specified in the execution JCL, the IMSCTRL macro uses the default IRLM name ('IRLM').

Batch Systems

To include the data sharing support of DBRC, you can specify DBRC=YES or FORCE in the IMSCTRL macro. For batch systems that do not use the IRLM, you can specify IRLM=N on the IMSCTRL macro. The batch JCL can then be changed to use the IRLM as required, using an EXEC parameter IRLM=Y.

Declaring Online Databases That Share Data

This section describes how to declare access for an online database and how to exclude a database from data sharing.

Declaring Access for Online Databases

You use the ACCESS keyword in the /START command to show how the subsystem requesting access to a database plans to use the database. The presence of any update intent means the whole system requires update access for that database. If any database PCB requires exclusive use, the whole system requires update access for that database. If no updating application programs need update access, you must find out what kind of read access is required. If the online system is to use block-level sharing, you usually declare read access.

The values and meaning of the ACCESS keyword are:

EX The named database is owned by this online system, and the system has exclusive access to the data. No restrictions exist on the access of any scheduled application program. Through the controlling action of DBRC, no other IMS system can access the data.

RO The named database can be shared with other IMS systems, but scheduled application programs in this online system can only have read-only access to the data. The data sharing can be at the database level or block level.

RD The named database can be shared with other IMS systems, but application programs can only be scheduled in this online system if they have read or read-only access to the data.

UP The named database can be shared with other IMS systems, and any scheduled application program can read the data or be eligible for update access to the data.

Other application programs can be scheduled in another IMS system to update this database if the online systems are participating in block-level sharing. Batch systems with update access can be authorized. A batch system can be authorized for read-only access for sharing at the database level.

If you plan to use the online version of the Database Image Copy utility, you do not need to specify ACCESS=UP. Although the utility requires a somewhat limited access while it is creating an image data set, the access and authorization are managed by DBRC.

Excluding a Database from Data Sharing

If you specify ACCESS=EX or leave the value blank, the database is not accessed concurrently by any other IMS subsystem. Register the database with DBRC, specifying a share level of zero.

Data Sharing System Definition

Databases whose data does not need to be shared with any other IMS system are owned by the IMS online system you define. If you do not want to use DBRC in controlling authorization to access these databases, don't register them.

GSAM and MSDB databases cannot be shared and should not be registered to DBRC.

Initializing DBRC Support

When planning for the use of DBRC, you must register databases and plan their recovery procedures. For details on how to use DBRC, see *IMS/ESA Operations Guide*.

Tailoring Execution JCL

When you are preparing a configuration that includes data sharing, you need to:

- Specify system data sets.
- Specify the databases with DISP=SHR.
- Specify the placement of database data sets and device choices to suit the physical paths to the data.
- Prepare IRLM procedures.

The system data sets that must be coordinated so that their status is common to all systems participating in data sharing are:

- RECON data sets
- Database data sets, with DISP=SHR specified
- IMS.ACBLIB, PSBLIB, and DBDLIB
- Libraries holding randomizing routines and DL/I exit routines

The system log is mandatory for any batch IMS system that has update intent against any database and operates with active DBRC. If the batch system only uses read or read-only access intent, the system log is not required.

You also need to set up IRLM procedures for your data sharing configuration. For information on preparing the IRLM procedures for a data sharing environment, see "Setting Up IRLM Procedures" on page 333.

Tailoring the Operating System

Because block-level sharing requires the use of the IRLM installed as an independent component under the MVS operating system, several tailoring actions must be performed. You need to:

- Coordinate VSAM data set definitions.
- Tailor the MVS system for IRLM.
- Define an IRLM if sysplex data sharing (for IRLM 2.1). See "System Initialization for IRLM" on page 335.

Coordinating VSAM Data Set Definitions with Share Options

When your database access method is VSAM, the declaration of the data set indicates to VSAM what degree of shared access is desired. The correspondence between the type of sharing and the SHAREOPTIONS parameter (in the DEFINE CLUSTER keyword) is shown in Table 38 on page 319.

Table 38. SHAREOPTIONS Parameter Specifications

SHAREOPTIONS Value	Type of Sharing
(1,3)	No sharing, single updater, or multiple readers
(2,3)	Single updater and multiple readers
(3,3)	Multiple updaters and multiple readers

For databases that are to participate in block-level sharing and use the VSAM access method, you must include the SHAREOPTIONS (3,3) parameter when defining the data sets. The RECON data set is also accessed as a KSDS and requires SHAREOPTIONS (3,3). For preparation of MVS/DFP Access Method Services statements that declare and catalog the share options, see *MVS/DFP Access Method Services for the Integrated Catalog Facility*.

When using IRLM, you must define SHARELEVEL=1 VSAM databases to VSAM with the appropriate SHAREOPTIONS required for block-level data sharing. This might require you to make a VSAM definition change.

Tailoring the MVS System for IRLM

Several other system initialization activities are required for the IRLM component:

- Defining the IRLM as an MVS subsystem
- Allowing for IRLM trace output
- Arranging for formatted dump output
- Defining an IRLM for sysplex data sharing

These activities are described in “System Initialization for IRLM” on page 335.

Monitoring and Tuning Considerations

This section identifies:

- What monitoring activities should be planned
- Performance factors and areas for tuning

The topics provided should be assessed based on how much your installation is using data sharing support. If you are only using database-level sharing for a gain in scheduling flexibility, you do not need to plan data gathering and analysis to the extent suggested. However, do not start any data sharing without a performance review and an analysis of the effect on your tuned systems.

Monitoring Systems That Share Data

Make allowances for additional monitoring when the IMS systems that previously executed in an independent fashion begin to use a shared data environment.

For dynamic monitoring, you must obtain feedback from active data sharing, and information on whether any immediate effects have occurred.

For daily monitoring, the status of the total workload processed should be tracked and response times studied for all critical transactions. You can use the historical trace of events recorded in the RECON data set to find out the intervals when shared data was active. The DBRC `LIST` command gives you a comprehensive

listing of events. For inter-MVS sharing, you need a listing from both processors. Events such as IMS system signon and time stamped database data set OPEN/CLOSE can be traced.

For detailed monitoring, you can use the MVS component trace for IRLM 2.1 to obtain a trace of the IRLM activity. This activity must be coordinated with the MTO and the system console operation. The MTO has to arrange for tracing to be controlled by the MVS `TRACE CT` command or `TRACE=YES` on the IRLM 2.1 irlmproc. `TRACE=YES` commands are entered at the system console. Tracing options are available for both dynamic lock control activity and IRLM communication cycle control.

Another approach is to use the IMS tracing facilities to provide raw data on lock activity. During online operation, specify with the `/TRACE` command the PI and LOCK options. (To direct the output to the system log and specify trace table size might require alteration of the OPTIONS control statement in the DFSVSMxx member of IMS.PROCLIB.) By using trace records from the system log, you can examine the pattern of obtaining and waiting for locks. If the data sharing configuration involves more than one online system that is sharing data needed by critical transactions, concurrent tracing from each IMS subsystem might be necessary.

Plan on detailed monitoring of the devices that are used for databases that are shared between IMS systems. You need to discover whether the I/O is evenly distributed and if contention exists along one access path. With shared data, the design factors that help performance, such as data set placement, device speed, and contention for a common part of the database, become even more important.

Performance Factors and Tuning Actions

Your primary performance-related activity when you are using data sharing is to monitor the effect upon a tuned IMS online system. The advantage of flexible scheduling of batch jobs must be balanced against any noticeable degradation in end-user service. If you predict high activity against a shared common database, those transactions accessing that database should be expected to have increased processing time and delays in response caused by IRLM communications.

The following tuning activities should be reviewed:

- “Limiting the Number of Processors with Concurrent Access”
- “Identifying Resource Contention Areas”
- “Tuning for Database I/O” on page 321
- “Tuning the IRLM” on page 321

Limiting the Number of Processors with Concurrent Access

Although an I/O configuration allows up to eight physical accesses to a single database data set, you should not have more than four. Because of contention for the DASD control unit, more than four physical accesses can result in unacceptable service times.

Another aspect of sharing that applies to online systems is that of distributing the overhead of the IRLM control. The cost of the communications between IRLM components is spread among transactions if a large number of regions is involved.

Identifying Resource Contention Areas

Several types of analysis can indicate potential resource contention, in conjunction with database administration:

- Examine the pattern of usage of the shared database records.

If a cluster of data is needed by application programs in two online systems, or if the number of physical blocks is small, contention to share the same resources is probable.

- Examine the pattern of database update commit points.

  A batch program accessing a shared database with extensive updating can result in an increased response time for online transactions if the resources are not released frequently enough. The batch program's need for locks also affects the IRLM storage utilization. The application program design might need to include increased use of CHKP calls so that database resources are released more frequently.

- Examine the impact of using the IRLM.

  The delays in obtaining global locks result in an increase in response time and increase the time IMS internal resources are being held by a transaction. Such increases can disturb the tuning stability of each online system taking part in the database sharing.

IRLM has a global deadlock manager that is arbitrarily designated as the IRLM in the group with the lowest IRLMID. This is determined in the startup proc. IRLMs in the group dynamically readjust this global manager identity as members join and leave the group. To improve performance, put the IRLM with the lowest ID on the fastest processor.

Tuning for Database I/O

It should not be necessary to alter the many performance attributes built into your database definition (DBD). Randomizing and blocking to take advantage of the distribution of data and storage device characteristics are more important than IRLM locking. A complex relationship exists between the effects of managing a large number of locks and of managing a smaller number with contention. Plan on concentrating your tuning actions on the IRLM cycle of control.

Tuning the IRLM

If the database I/O subsystem and physical record distribution are optimized, the primary tuning parameter DEADLOK in the MVS `START` command controls the IRLM deadlock detection. Frequent local deadlock detection, using a low value for the first of the DEADLOK values, increases IRLM processing devoted to local deadlock detection. Global deadlock detection also increases processing when it is performed, and it requires IRLM-to-IRLM communication. However, frequent deadlock detection reduces the length of time that application programs are left waiting in a deadlock.

Administering Sysplex Data Sharing

This section assumes you are familiar with the nonsysplex data sharing environment described in previous sections of this chapter. The information that has been presented about nonsysplex data sharing applies equally to sysplex data sharing, except as noted in this section.

This section also assumes you are familiar with System/390 MVS sysplex concepts and terminology.

Related Reading: For information on MVS sysplex, see *Sysplex Overview.*

Sysplex Data Sharing Concepts and Terminology

When multiple IMS systems share data across **more than two** MVS images, it is called *sysplex data sharing*.

Sysplex Data Sharing

Sysplex data sharing is possible with up to 32 IMS subsystems. You can, for example, have:

- Eight MVS images, each with four IMSs
- Four MVS images with one IMS each, and one MVS image with up to 28 additional IMSs

Figure 49 shows a common configuration for sysplex data sharing.

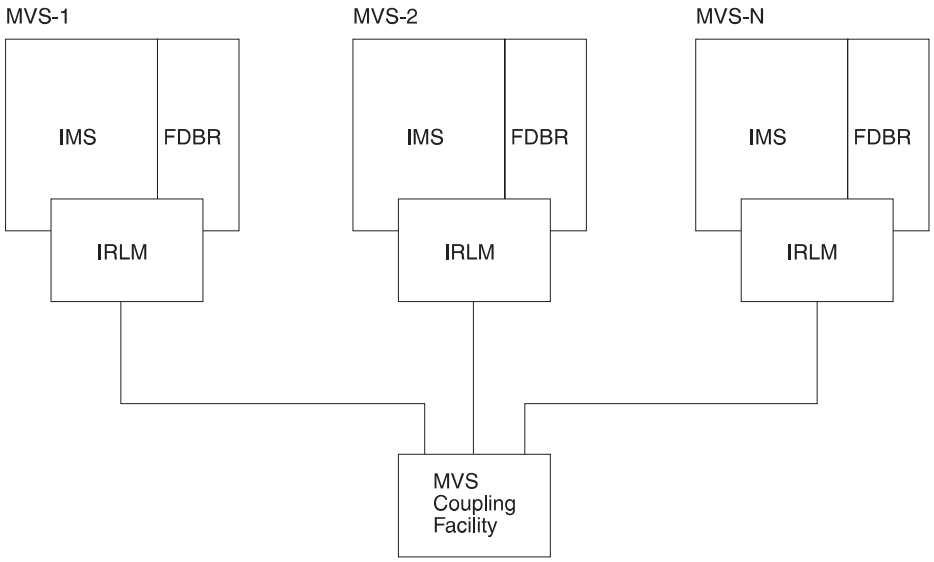


Figure 49. Sysplex Data Sharing

Each MVS image involved in sysplex data sharing must have at least one IRLM (IRLM 2.1) and a corresponding Fast DB Recovery (FDBR) region. Notice that each IRLM in the figure is connected to a coupling facility. The coupling facility, as explained later, is used to maintain data integrity across IMS systems that share data.

Related Reading: For more information on the coupling facility, see:

- *MVS/ESA SP V5 Programming: Sysplex Services Guide*
- *MVS/ESA SP V5 Programming: Sysplex Services Reference*

For more information on fast database recovery see “Fast Database Recovery” on page 325.

Buffer Invalidation

When a database block or CI is updated by one IMS, IMS must make sure all copies of this block or CI in other IMS buffer pools are marked invalid.

To do this, the updating IMS issues an invalidate call. This call notifies the coupling facility that the content of a buffer has been updated and that all other copies of the buffer need to be invalidated. The coupling facility then invalidates the buffer for each IMS that registered an interest in the buffer. This process is called *buffer invalidation*. Whenever IMS tries to access a buffer, it checks to be sure the buffer is still valid. If the buffer is invalid, IMS rereads the data from DASD. Data integrity is thereby maintained.

Buffer invalidation works in all IMS sysplex database environments: DB/DC, DBCTL, and DB batch. In the sysplex environment, IMS supports buffer pools for VSAM, VSAM hiperspace, OSAM, and OSAM sequential buffering buffers.

Data Sharing Groups

As with nonsysplex data sharing, the concept of a *data sharing group* applies. Figure 50 shows a sample data sharing group.

The IMS subsystems are members of this group, and they share:

- Databases
- A RECON dual copy data set
- One or more coupling facilities
- A single IRLM lock table structure in the coupling facility (hereafter called an IRLM structure)
- OSAM and VSAM buffer invalidate structures in the coupling facility (hereafter called an OSAM or VSAM structure)

Fast database recovery regions are not shared.

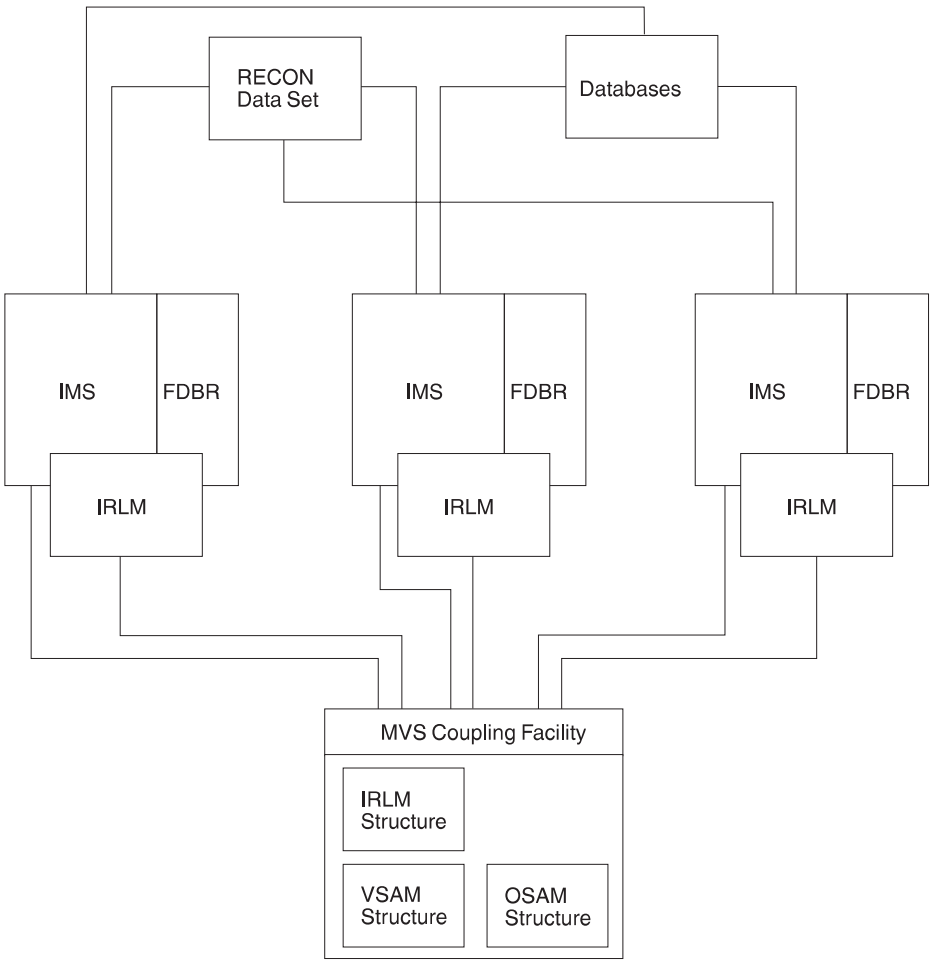


Figure 50. Sample Data Sharing Group

Communication in the data sharing group is through the IRLMs connected to the coupling facility. Without a coupling facility, no valid sysplex data sharing environment exists; the IRLM does not grant global locks, even for nonsysplex data

sharing. IRLM 2.1 requires a coupling facility for inter-MVS data sharing. If a coupling facility is not available, IRLM 2.1 only allows intra-MVS data sharing.

IMS connects to the structures in the coupling facility. Sysplex data sharing uses the OSAM and VSAM structures for buffer invalidation. This differs from nonsysplex data sharing, which does buffer invalidation using broadcasts (notifies) to each IMS. Sysplex data sharing uses the IRLM structure in the coupling facility to establish and control the data sharing environment.

A data sharing group is defined using the CFNAMES control statement, as described in "Defining Sysplex Data Sharing" on page 329. The IRLM, OSAM, and VSAM structures are named on the CFNAMES control statement.

Coupling Facility

Figure 50 on page 323 shows the coupling facility and the three structures in it that are used for IMS sysplex data sharing.

Although the figure shows a single coupling facility, more than one are possible. Additional coupling facilities can be defined for backup. Or, to increase throughput, structures can be split across coupling facilities; the IRLM structure, for example, can be put on one coupling facility and OSAM and VSAM structures on another.

The OSAM and VSAM structures, as mentioned earlier, are used for buffer invalidation. For each block of shared data read by any IMS connected to the coupling facility, an entry is made in the OSAM or VSAM structure. Each entry consists of a field for the buffer ID (known to MVS as the resource name) and 32 slots. The slots are for IMS subsystems to register their interest in an entry's buffer. This makes it possible for as many as 32 IMS subsystems to share data. Note that the sysplex data sharing limit of 32 IMSs is the number of IMSs that can connect to a structure; it is not the number of IMSs that are running.

The IRLM structure is used to establish the data sharing environment. For a data sharing group, the first IMS to connect to an IRLM structure determines the data sharing environment for any IMS that subsequently connects to the same IRLM structure. When identifying to the IRLM, IMS passes the names of the coupling facility structures specified on the CFNAMES control statement, plus the DBRC RECON initialization timestamp (RIT) from the RECON header. The identify operation fails for any IMS not specifying the identical structure names and RIT as the first IMS.

If a structure fails, IMS tries to rebuild it. If IMS is unable to rebuild it or if the connection to a structure is lost, IMS quiesces data sharing transactions and stops data sharing. After the error is resolved, IMS tries to reconnect to the structure. If the reconnect succeeds, IMS continues processing data sharing transactions. If the reconnect fails, data sharing remains stopped and IMS waits to reconnect until it is again notified that coupling facility resources are available.

Defining a CFRM Policy for Shared Queues

If you are operating in a shared-queues environment, your MVS system programmer must define a Coupling Facility Resource Management (CFRM) policy. A portion of the CFRM policy defines coupling facility structures for up to four shared queues:

- Primary message queues
- Overflow message queues
- Primary EMH queues

- Overflow EMH queues

The overflow queue structures are optional. The EMH queues are only required if Fast Path is used.

The structure names must be consistent in all of the following places:

- The CFRM policy
- The IMS procedure
- The CQS PROCLIB member

Defining an MVS LOGR Policy

If you are operating in a shared-queues environment, your MVS system programmer must define an MVS Logger (LOGR) policy. The log stream names defined in this policy must be the same as the log stream names defined in the CQS PROCLIB member.

Each structure pair in the coupling facility has one MVS log stream.

Defining an IMS XCF Group

All IMS subsystems in the shared queues group must be part of the same XCF group. This group should not include any of the CQS subsystems.

Fast Database Recovery

This section describes how Fast DB Recovery functions in an IMS sysplex environment.

The Fast DB Recovery region is executed by the IMS SYSGEN-supplied cataloged procedure. You must start it after you start the IMS subsystem that it tracks.

A Fast DB Recovery region tracks a single IMS subsystem, so a system administrator must establish a Fast DB Recovery region for each instance of IMS in an XCF group.

Fast DB Recovery must have access to the following components in order to perform the tracking (surveillance) functions that enable it to recover an affected database during a failure.

- RESLIB
- ACBLIB
- MODBLKS
- MODSTAT
- OLDS and WADS
- Recovery data set (RDS) for checkpoint information
- RECON data set

Once the Fast DB Recovery region is established, it tracks an IMS subsystem in the same XCF group. The Fast DB Recovery region monitors the log records and database activity for the IMS subsystem. Any activity that involves an MSDB is ignored. A DEDB that is specified as nonsharing (SHARELVL=0|1) can be tracked by the Fast DB Recovery region.

As it monitors IMS, Fast DB Recovery updates its control blocks to reflect the changing status of the tracked subsystem. It continually checks its surveillance mechanism and log records for signs that the tracked IMS is failing.

XRF and Sysplex Data Sharing

XRF can be used in a sysplex data sharing environment only if the coupling facility is available and connected when the alternate system is started.

Restriction: You cannot use Fast DB Recovery for XRF subsystems.

Sample Sysplex Data Sharing Configurations

Figure 51 through Figure 55 on page 328 shows several possible configurations for sysplex data sharing. The DBRC RECON data set and the IMS database are not shown in these figures but, as in Figure 50 on page 323, RECON and the database are shared by the IMSs in the sysplex data sharing group.

Three Structures on One Coupling Facility

The following configuration shows multiple MVS images running in the sysplex data sharing environment using IRLM 2.1. IMS V6 is running on each MVS image. One coupling facility is being used. The IRLM, OSAM, and VSAM structures are on the same coupling facility. The coupling facility could have shown the IRLM and either VSAM or OSAM.

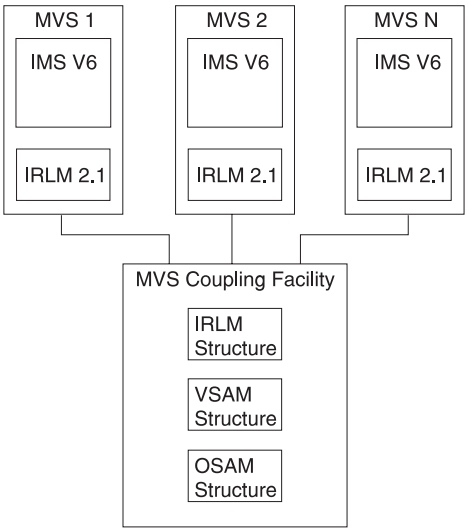


Figure 51. Three Structures on One Coupling Facility

Three Structures on Two Coupling Facilities

The following configuration shows multiple MVS images running in the sysplex data sharing environment using IRLM 2.1. IMS V6 is running on each MVS image. Two coupling facilities are being used. The IRLM structure is on coupling facility 1. The OSAM and VSAM structures are on coupling facility 2.

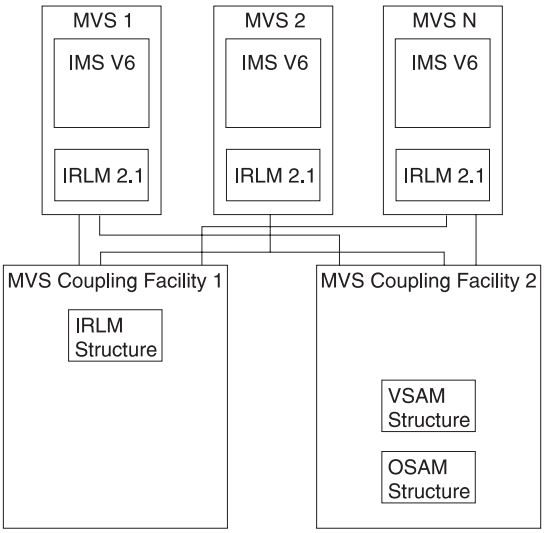


Figure 52. Three Structures on Two Coupling Facilities

Three Structures on One Coupling Facility with Backup

The following configuration shows multiple MVS images running in the sysplex data sharing environment using IRLM 2.1. IMS V6 is running on each MVS image. Two coupling facilities are being used. The IRLM, OSAM, and VSAM structures are on coupling facility 1. Coupling facility 2 is used as a backup so that if coupling facility 1 fails, structures can be rebuilt on coupling facility 2.

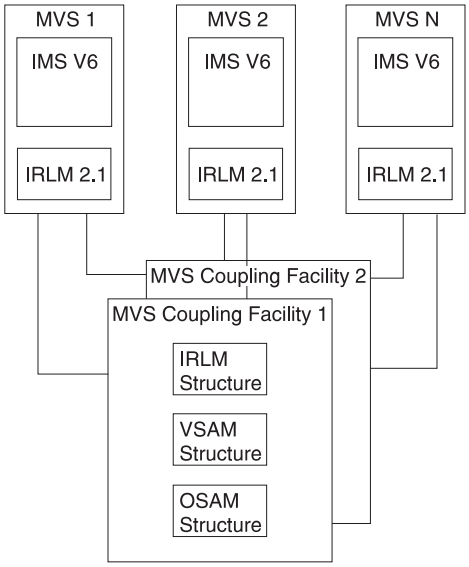


Figure 53. Three Structures on One Coupling Facility with Backup Coupling Facility

Three Structures on One Coupling Facility with Backup (XRF)

The following configuration shows six MVS images running in the sysplex data sharing environment using IRLM 2.1. Three active (XRF) 6.1 IMSs are running, and each active IMS has an alternate system. The IRLM, OSAM, and VSAM structures are on coupling facility 1. Coupling facility 2 is used as a backup so that if coupling facility 1 fails, structures can be rebuilt on coupling facility 2.

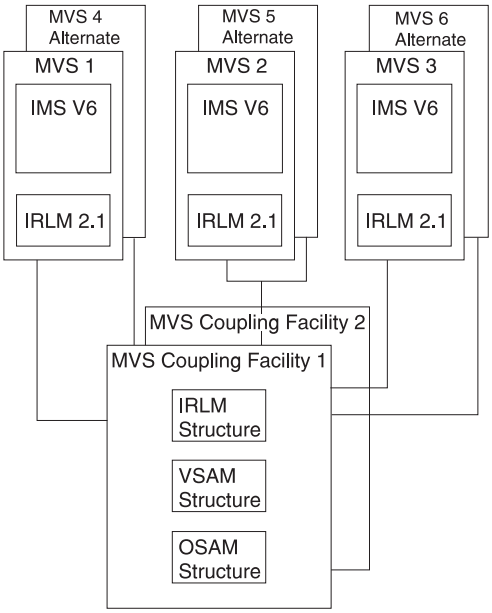


Figure 54. Three Structures on One Coupling Facility with Backup Coupling Facility (XRF Environment)

One Structure on One Coupling Facility

The following configuration shows multiple MVS images running in a data sharing environment using IRLM 2.1. IMS V6 is running on each MVS image. One coupling facility is being used. The IRLM structure is on the coupling facility. No OSAM or VSAM structures are on the coupling facility. This configuration results in a sysplex data sharing environment using the notify protocol for buffer invalidation.

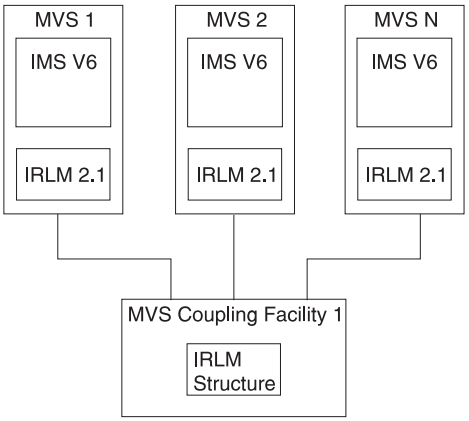


Figure 55. One Structure on One Coupling Facility (Results in Sysplex Data Sharing)

When to Use Sysplex Data Sharing

The two primary reasons for moving to a sysplex data sharing environment involve cost and availability. With System/390 MVS sysplex data sharing, you can take advantage of the lower cost of the new CMOS-based hardware, spreading the workload to multiple CPUs while maintaining one logical view of your databases; this can be done without making any application software changes. Availability in this environment is improved, because if one CPU in a sysplex is lost, the workload can be moved to the remaining CPUs.

Converting Batch Jobs to BMPs

The number of connections that can be made to each coupling facility structure is 32. Each IMS connected to an OSAM or VSAM structure uses one connection or slot. Therefore, every batch job running in data sharing mode uses a slot. This means any jobs in the same data sharing group (started after the combination of active online and batch jobs) that exceed 32, are unable to run. If you convert these batch jobs to BMPs, jobs are more likely to be able to run because BMP jobs run under the online region coupling facility connection.

Related Reading: For more information on converting a batch job to a BMP, see *IMS/ESA Application Programming: Design Guide*.

Defining Sysplex Data Sharing

System definition and tailoring are basically the same for sysplex and nonsysplex data sharing except that:

- Coupling facility structure names must be specified for sysplex data sharing.
- If you are migrating to sysplex data sharing, VTAM CTC definitions for IRLM 2.1 are no longer required. You can remove the definitions from the IRLM procedure.

See previous sections in this chapter for information on system definition and tailoring. These previous sections apply to both data sharing environments. The rest of this section describes how the definition of sysplex data sharing differs from the definition of nonsysplex data sharing.

Three coupling facility structure names must be specified for sysplex data sharing. The names are for the:

- IRLM structure (the IRLM lock table name)
- OSAM structure
- VSAM structure

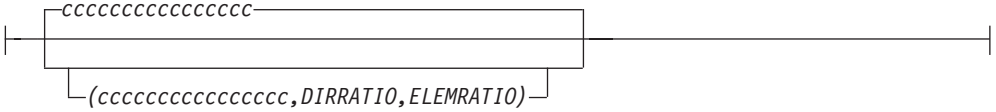
These structure names are specified in the CFNAMES control statement. In an online IMS system, the CFNAMES control statement is placed in the IMS.PROCLIB data set in member DFSVSMxx. In a batch environment, the CFNAMES control statement is placed in the DFSVSAMP data set. The specified structures define the data set group. Structure names are originally defined in the MVS coupling facility policy. At run time, CFNAMES lets you select which of these previously defined structures you want to use.

The CFNAMES control statement has the following format:

frag1:

frag2:

frag3:



The CFIRLM name is required. The CFVSAM and CFOSAM parameters are required but can contain null values. The structure names that you specify can be 1 to 16 characters. If CFNAMES control statements beyond the first one are encountered, they are not processed and an error message is issued. If parameters on the CFNAMES control statement are duplicated, only the first one is used and an error message is issued.

Example of CFNAMES Control Statement

A sample VSPEC member with a CFNAMES statement is as follows:

```
CFNAMES,CFIRLM=LT01,CFOSAM=OSAMSESXI,CFVSAM=VSAMSESXI
8192,4
16348,4
1024,8
IOBF=(16348,4,N,N)
IOBF=(6144,4,N,N)
IOBF=(8192,4,N,N)
OLDSDEF OLDS=(00,01,02,03),MODE=SINGLE,BUFNO=20
WADSDEF WADS=(0,1)
OPTIONS,DLOG=ON,SCHD=ON,LATC=ON,STRG=ON,DUMP=YES
OPTIONS,DISP=HIGH,LOCK=HIGH,DL/I=HIGH
```

The following sample JCL shows the definition of IMS and IRLM structures (policy) in the Couple data set for CFRM:

```
//HPC$PLCY JOB MSGCLASS=A,REGION=2000K,CLASS=K,MSGLEVEL=(1,1)
//POLICY EXEC PGM=IXCM2APU
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN  DD *
 DATA TYPE(CFRM)
 DEFINE POLICY NAME(CONFIG01) REPLACE(YES)
   STRUCTURE NAME(LT01)
             SIZE(32768)
             REBUILDPERCENT(50)
             PREFLIST(LF01,LF02)
   STRUCTURE NAME(LT02)
             SIZE(32768)
             PREFLIST(LF01,LF02)
   STRUCTURE NAME(LT03)
             SIZE(4)
             PREFLIST(LF01,LF02)
   STRUCTURE NAME(LT04)
             SIZE(9000)
             PREFLIST(LF01,LF02)
   STRUCTURE NAME(OSAMSESXI)
             SIZE(2048)
             PREFLIST(LF02,LF01)
             REBUILDPERCENT(50)
```

```
STRUCTURE NAME(VSAMSESXI)
          SIZE(2048)
          PREFLIST(LF02,LF01)
          REBUILDPERCENT(50)
STRUCTURE NAME(IXCLINKS)
          SIZE(4000)
          PREFLIST(LF01,LF02)
```

IRLM

In a sysplex data sharing environment, you must use the IRLM. In addition, you must make sure IMS is connected to the correct IRLM and that the IRLM is connected to the correct coupling facility.

If you are sharing data using IRLM 2.1, a default IRLM structure name is specified as part of the IRLM startup procedure. Connection of the IRLM to the coupling facility proceeds as follows:

- If IMS is started and no CFNAMES control statement is submitted, the IRLM connects to the coupling facility using the default name.
- If IMS is started and the CFNAMES control statement specifies only an IRLM structure name, the IRLM connects to the coupling facility using the specified IRLM structure name. When CFNAMES specifies only the IRLM structure name, the environment defaults to data sharing using the notify protocol for buffer invalidation.
- If IMS is started and the CFNAMES control statement specifies an IRLM structure name and OSAM or VSAM structure names, what happens depends on whether this is the first IMS in the data sharing group to identify to the IRLM.
  - If this is the **first** IMS to identify, the IRLM connects to the coupling facility using all specified names.
  - For any IMS that **subsequently** identifies itself to an IRLM using the same IRLM structure name, the IRLM determines if the OSAM and VSAM structure names and the DBRC RECON initialization timestamp match those it already knows about. If so, connection to the coupling facility proceeds. If not, the identify operation is rejected and IMS initialization fails.

    The important point to understand here is that with IRLM 2.1 data sharing, the first IMS in a data sharing group to identify itself to an IRLM sets the operational environment for all other IMSs trying to identify themselves to an IRLM as part of that data sharing group.

OSAM

The buffer invalidation process supports OSAM sequential buffers.

VSAM

The buffer invalidation process supports VSAM hiperspace buffers.

Calculating the Size of Coupling Facility Structures

The size of structures in the coupling facility is determined using an MVS formula. You must calculate certain IMS values in this formula. This section shows you how to calculate them.

A general recommendation for calculating structure sizes: use transaction rates for peak processing periods. Doing so allows you to avoid resetting structure sizes. Overestimating sizes causes no problems, while underestimating sizes can cause abends.

IRLM Structure

For installation planning purposes, calculate the IRLM structure size as follows:

1. Add all the ECSA storage used for IRLM control blocks for all the IRLMs in the data sharing group. ECSA storage is specified in one of the following places:
   - MAXCSA parameter on the IRLM procedure (DXRJPROC)
   - IRLM RGNSIZE parameter, if PC=YES on the IRLM procedure (DXRJPROC).

   Related Reading: See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on DXRJPROC.
2. Divide the result by 2.
3. Round up so that the value is a power of 2, such as 8 MB, 16 MB, 32 MB, and 64 MB.

A sample calculation follows. ECSA storage is 6 MB. Ten IMSs, each with its own IRLM, are in the data sharing group. A rough estimate of the required storage follows:

Calculating IRLM Structure Size

(6 MB x 10)/2 = 30 MB rounded up to 32 MB

After setting the initial value, monitor the use of the IRLM structure using IRLM messages, some of which suggest that you increase the size of the structure. You will begin to receive IRLM messages when 50% of the current IRLM structure is used.

OSAM and VSAM Structures

For sysplex data sharing, the size of the structure required by each access method depends on the number of buffers defined to the access method by each IMS in the data sharing group. The buffers for all IMS control regions and batch jobs that are data sharing and registered to the IRLM need to be counted. Here are formulas for doing this:

Count of OSAM Buffers

OSAM buffer count = #osambfrs/IMS_n + #osambfrs/IMS_{n+1} + ... # osambfrs/IMS_{n+x}

Count of VSAM Buffers

VSAM buffer count = #vsambfrs/IMS_n + #vsambfrs/IMS_{n+1} + ... #vsambfrs/IMS_{n+x}

An OSAM structure may optionally be used for caching data. If data caching is not used, the structure contains only directory entries. If data caching is used, the structure contains directory entries and data elements. In this case the size of the structure must allow for data elements. OSAM data is stored in the structure as multiples of 2 KB data elements. Using data caching requires a directory-to-element ratio to be specified. This ratio causes the structure to be subdivided with directory entries and data elements. The ratio is specified with the CFOSAM= keyword on the CFNAMES parameter statement.

Related Reading: For more information on the ratio is specified with the CFOSAM= keyword on the CFNAMES parameter statement, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The OSAM buffer count must include any sequential-buffering buffers that are defined. The VSAM buffer count must include any hiperspace buffers that are defined.

MVS Formula: After you have calculated the OSAM buffer count, put the result in the TDEC field of the MVS cache structure size formula. Then do the same for the VSAM buffer count. (The calculation is done separately for each access method.) TDEC is the count of IMS buffers (either OSAM or VSAM).

Related Reading: For information on OSAM Database Coupling Facility Caching, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Setting Up IRLM Procedures

For each IRLM component that can be activated, you need a procedure that is invoked when the system console operator enters the MVS `START` command. An example of one such procedure is DXRJPROC, described in *IMS/ESA Installation Volume 2: System Definition and Tailoring* . If you plan to use more than one IRLM procedure on a single processor, include additional members in IMS.PROCLIB or SYS1.PROCLIB. Additional IRLMs require a unique identifier and any desired changes in parameter values. The procedure name cannot be the same as the IRLM subsystem names established for MVS.

For each IRLM component, the procedure parameters specify:

- The component's identifying name
- The scope of its control
- Sysplex data sharing information (for IRLM 2.1)
- IRLM actions in failure situations
- The amount of virtual storage
- Performance factors

The PARM1= and PARM2= positional parameters for the region that executes in support of block-level sharing are shown in Table 39.

Table 39. Categories and Purpose of IRLM Region Parameters

Category	Parameter (PARM1 and PARM2)	Used to specify...	Applicable IRLM Version
Component name	IRLMID	An identifier for this IRLM.	2.1
	IRLMNM	The MVS subsystem name.	2.1
Scope of control	SCOPE	Inter-MVS communication or intra-MVS control.	2.1
Data sharing	IRLMGRP	The name of the XCF group.	2.1
	MAXUSRS	The maximum number of users in the group.	2.1
	LOCKTAB	The IRLM structure to be used by the group.	2.1
Storage	MAXCSA	The maximum amount of CSA to be used.	2.1
	PC	Specify control blocks in private storage.	
Performance	DEADLOK	The deadlock detection timing.	2.1
Tracing	TRACE	Trace types DBM, XCF, and SLM turned on at IRLM initialization (EXP, INT, and XIT are always on).	2.1

Identifying the IRLM

The IRLMID parameter specifies a decimal number so that the IRLM can be identified. For IRLM 2.1 (sysplex data sharing), this number is from 1 to 255. The IRLMID gets converted to a hexadecimal equivalent. A unique number must be assigned for each IRLM. IRLM error and status messages include the subsystem name and ID of the IRLM that issued the message.

The IRLMNM parameter specifies the 1- to 4-byte MVS subsystem name that is to be assigned to this IRLM. Unless more than one IRLM is used at the same time on the same system, you can use “IRLM” as the subsystem name in each operating system. When IRLMs are used concurrently on a system, the subsystem names must be unique. For a description of how to define a subsystem to MVS, see *MVS/ESA Version 4 System Modifications*.

Specifying the IRLM Scope

You need to specify what kind of data sharing control is required of the IRLM component–whether inter-MVS or intra-MVS sharing is to be used. The SCOPE parameter accomplishes this. SCOPE=GLOBAL specifies inter-MVS sharing is to be controlled. This is sharing of resources among multiple systems.

SCOPE=LOCAL limits data sharing to intra-MVS sharing. This is sharing of resources among IMS subsystems in a single processor with a single IRLM. If a coupling facility is not available, SCOPE=LOCAL is required. SCOPE=NODISCON specifies inter-MVS sharing is to be controlled with special DISCONNECT rules. SCOPE=GLOBAL or NODISCON requires that a coupling facility be defined.

Limiting IRLM Use of CSA

The MAXCSA parameter specifies the maximum amount of CSA the IRLM is to use for its lock structures. The amount must be specified as a 1- to 3-digit decimal number from 1 to 999. This number indicates the multiple of 1 MB of extended CSA (ECSA) storage. For example, a specification of 3 states that the IRLM can use 3 MB of ECSA.

Use the PC parameter to reduce the CSA requirements. Specifying PC=YES causes the IRLM lock structures to reside in the IRLM address space. In this case MAXCSA is ignored but must be non-zero. If the IRLM is to operate with PC=YES, the storage required for lock structures is placed in extended private storage. With PC=NO, the control blocks are in the extended common storage area (ECSA).

Attention: IRLM uses CSA/ECSA for processing.

IRLM Performance Control

The DEADLOK parameter specifies a cycle of control. You specify two numbers for the parameter, both specified as 1 to 4-digit numbers between 1 to 9999. For the first value, you specify, in seconds, the interval that should elapse between IRLM local deadlock cycles. The second value specifies how many local deadlock cycles are to occur before global deadlock detection is performed.

In a data sharing environment, the participating IRLMs synchronize their DEADLOK parameters to the highest values provided. You should make this parameter the same for each IRLM in the data sharing group.

System Initialization for IRLM

The following sections describe the system initialization activities required for the IRLM component.

Defining an IRLM as an MVS Subsystem

Your installation must assign a 4-byte subsystem name. You can use “IRLM” as the name, which assists the system operator in recognizing system messages sent to the console. This name must be unique to the MVS system. If more than one IRLM subsystem is to run in an operating system, the names must be different.

An additional identification for each IRLM is a number in the range 1 to 255. This is required to internally differentiate between the IRLMs. For example, two processors that are to share data could have IDs of 1 and 2. Status and error messages sent to the system console operator contain the IRLM subsystem name and ID. Two IRLMs executing in a single MVS system, or in different MVS systems which are members of the same data sharing group, require different ID numbers.

Another consideration for the IRLM subsystem name is that it must not be the same as that used for the startup procedure. This procedure is a member of SYS1.PROCLIB and is used by the operator in the `START` command. A unique `START` procedure must exist for each IRLM subsystem. The procedure library members are also distinguished by the different ID numbers and other control parameters.

The execution of the IRLM requires that the subsystem be added to the MVS program properties table (PPT).

Related Reading: For more information on adding IRLM to the PPT, see *IMS/ESA Installation Volume 1: Installation and Verification*.

The system initialization routine for the subsystem must be null.

Related Reading: For more information on defining the IRLM as a subsystem and adding an entry to the subsystem name table, see *MVS/ESA System Programming Library: Initialization and Tuning*.

Allowing for IRLM Trace Output Printing

To provide a trace of the activity occurring in the IRLM subsystem, you use the MVS Component Trace facility or the TRACE=YES option of the IRLM procedure (DXRJPROC).

Related Reading: See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on DXRJPROC.

CTRACE Records: IRLM 2.1 produces trace output in MVS component trace (CTRACE) format. This allows you to use IPCS CTRACE format, merge, and locate routines to process the buffer data. Both the IRLM trace buffers in the dump and external writer data set can be formatted using IPCS. The IRLM trace formatting load module (DXRRLFTB) and buffer find routine load module (DXRRL186) must be available for IPCS.

Arranging for Formatted Dump Output

IRLM uses the SDUMP facility of MVS. The offline printing of dump output is done using MVS print dump.

Related Reading: For more information on SDUMP, see *IMS/ESA Installation Volume 1: Installation and Verification*.

Defining an IRLM for Sysplex Data Sharing (IRLM 2.1)

If you are implementing a sysplex data sharing environment using IRLM 2.1, you need to define the data sharing group to which the IRLM belongs, the lock structure to be used by that group, and the maximum number of users in that group.

Related Reading: For more information on a sysplex data sharing environment, see “Administering Sysplex Data Sharing” on page 321.

Defining the Data Sharing Group

For an IRLM to belong to a data sharing group, you must specify the name of the data sharing group and the name of the IRLM structure in the IRLM startup procedure. All IRLMs in this group can share the same data. Each IRLM in the group must:

- Have a unique IRLMID
- Specify the same data sharing group name using the GROUP parameter
- Specify the same lock structure using the LOCKTABL parameter

Although you can specify these definitions on the IRLM startup procedure, the recommended method is to define them using the CFNAMES control statement. See “Defining Sysplex Data Sharing” on page 329. The CFNAMES control statement defines the data sharing group for a sysplex data sharing environment. If you do not use the CFNAMES control statement, the IRLM structure name is pulled from the IRLM startup procedure.

Defining the Lock Structure

Each IRLM participating in a sysplex data sharing group (specifying the same XCF name on the GROUP=IRLMDS parameter) must specify the same lock structure. You specify the lock structure name using the LOCKTABL parameter.

Defining the Number of Users in a Data Sharing Group

You need to specify a maximum number of users for the data sharing group. You do this by specifying a value from 2 to 32 on the MAXUSRS parameter.

For More Information

For more information on data sharing, see the following publications:

- *IMS/ESA Operations Guide*, which describes operations and recovery in a data sharing environment.
- *IMS/ESA Sample Operating Procedures*, which contains detailed procedures for operations and recovery in a data sharing environment.
- The various reference manuals in the IMS library, especially *IMS/ESA Installation Volume 2: System Definition and Tailoring*, *IMS/ESA Messages and Codes*, *IMS/ESA Failure Analysis Structure Tables (FAST) for Dump Analysis*, and *IMS/ESA Diagnosis Guide and Reference*.

Chapter 11. Administering the IMS Spool API

This chapter provides design and operational advice for the use of the IMS DL/I Spool application program interface (API) and provides details for using the IMS Spool API to increase the access of IMS application programs to advanced printing capabilities.

IMS provides an expansion of the DL/I application program interface that allows application programs to interface directly to JES and to create print data sets on the JES spool. These print data sets are then made available to print managers and spool servers to serve the needs of the application program.

In this Chapter:

Design and Operational Considerations

This section introduces IMS support of IMS Spool API.

Native IMS Terminal Support

IMS supports a set of terminals and printers that are associated with the IMS control region through the IMS systems generation process and ETO. For most environments, a message destined for IMS-supported terminals is placed in the IMS message queue for intermediate storage. When the terminal can receive the message, IMS retrieves the message from the IMS message queue and sends the message to the device. If the VTAM session is lost during message transmission, or if a transmission error occurs, IMS places the message back on the IMS message queue for transmission at a later time. Because the IMS message queue is a recoverable resource, IMS can provide message integrity and recover from media failures associated with the IMS message queue.

Figure 56 on page 338 shows a simplified diagram of the IMS environment. The figure shows terminal devices controlled by the IMS control region, the IMS message queue, and a message processing region. The DL/I pre-processing routines and the services of the IMS control region provide both terminal and database services. An application program sending output messages to a printer can identify the printer through the DL/I CHNG call and insert messages to the printer through the DL/I ISRT call. IMS places these messages in the IMS message queue until they are committed by the application program. After the calls are committed, IMS delivers the messages to the designated printer.

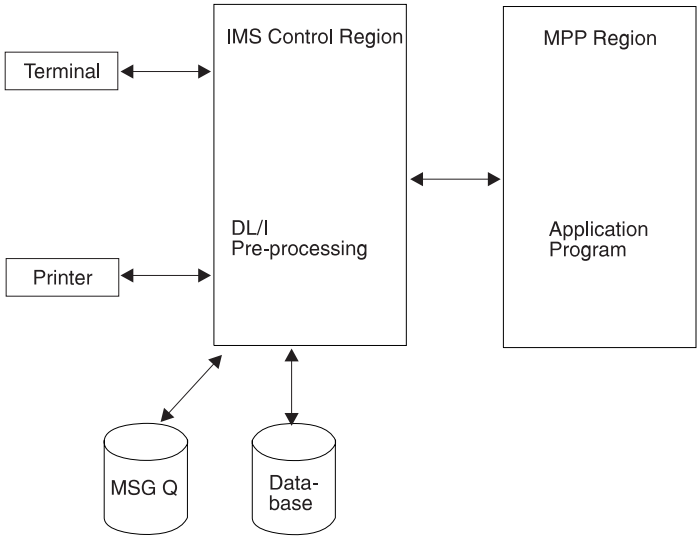


Figure 56. Current IMS Environment

Application Requirements

IMS application programs can send messages to IMS native printers and to printers that are capable of printing high-quality text, special fonts, bars, and graphs. Application programs that create data streams for these printers and send data to these printers use the job application subsystem IMS Spool API. The IMS Spool API passes the print data sets to IBM's Advanced Function Printing (AFP) services or to an OEM print server based on data set class, remote destination, size (segment sizes of up to 32 KB are common), or other characteristics.

Applications creating JES print data sets from IMS application regions require reasonable performance, a simple interface, and flexibility in program usage. The IMS Spool API interface extensions attempt to provide for these needs. The DL/I application interface supports the creation of JES print data sets. The techniques for creating these JES print data sets are similar to the techniques used for sending messages to native IMS printers.

IMS uses several MVS services to support IMS Spool API. These MVS services introduce performance and availability considerations that IMS and its application programs have not had to deal with previously, ranging from additional processor requirements for data set options parsing to lack of sync point support by the JES subsystem. The IMS Spool API environment shown in Figure 57 on page 339 is designed to address these issues. Because the print data sets can be very large and because using the message queue as intermediate storage adds processing overhead without providing additional message integrity, most IMS Spool API execution is performed in the IMS dependent region. However, this asynchronous message technique also presents problems in relating application program errors that result in message delivery problems.

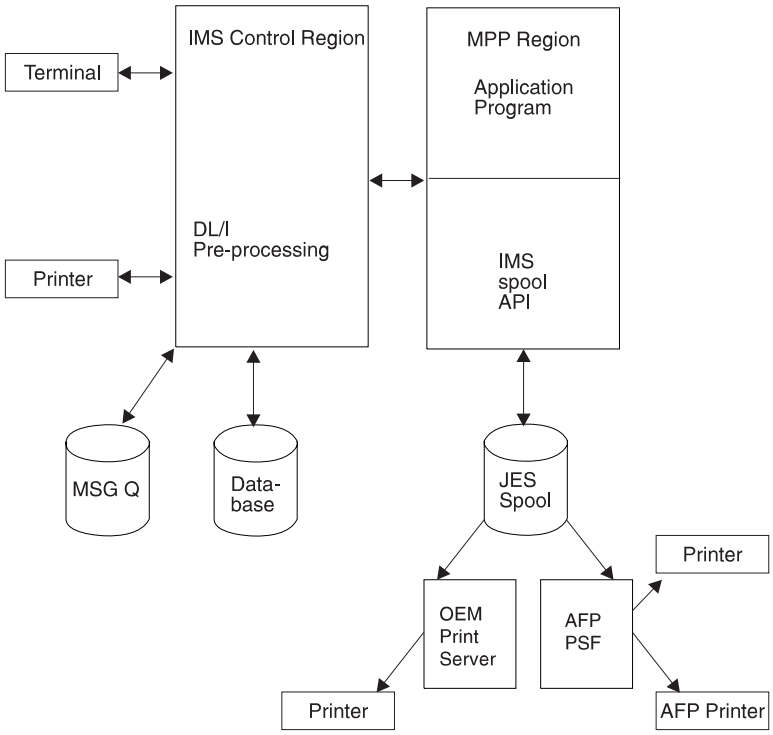


Figure 57. IMS Environment with Spool API

Note: Do not confuse the support for creating JES print data sets on the IMS Spool API with the IMS terminal support for spool lines defined as UNITYPE=SPOOL in the IMS systems generation process. Terminal support for spool lines is completely unrelated to support for JES print data sets.

The IMS Spool API as a Data Manager

The IMS Spool API controls the input, execution, and output of jobs on the MVS system. Because the IMS Spool API is at the JOB or time sharing (TSO) user boundary, JES creates special considerations for IMS environments where each print data set is a logical message and is not part of or associated with a job boundary.

A data manager associated with IMS provides the DL/I application program interface, which provides temporary buffering of data with the sync point process. The sync point process commits the changes for a unit of work if the unit of work terminates normally, or backs out the changes if the unit of work terminates abnormally. The IMS Spool API does not use the sync point process, but it can back out the effect of changes made by an abending unit of work.

The IMS Spool API does not recover data as does the IMS Transaction Manager or the IMS Database Manager. Because the IMS Spool API does not log changes to the spool data set, data on the spool can be lost in a restart if the spool device finds data errors.

Print Data Set Characteristics

You can create a JES print data set in these ways:

- Allocate a print data set by the SYSOUT= specification on a JCL statement

- Dynamically allocate a print data set using the MVS dynamic allocation support for SVC99

When a print data set is allocated, the characteristics of the print data set can be defined or allowed to default. Some examples of these print data set characteristics are FORMS, COPIES, DESTINATION, and WRITER. You can specify some of these output characteristics with the DD card, or you can associate the print data set with a set of print data set characteristics defined by an OUTPUT JCL statement. JES provides a way to dynamically define these print data set descriptors using system services known as dynamic output or SVC109 processing. When used with dynamic allocation, dynamic output processing can associate system output or SYSOUT data sets with a set of dynamically built OUTPUT descriptors.

Related Reading: For more information on MVS services for dynamic output (SVC109) for print data sets, see *MVS/ESA System Programming Library: Application Development Guide*.

You can build print data set descriptors for IMS Spool API in three ways:

- The change (CHNG) call
- The set options (SETO) call
- The output DD statement

The Change (CHNG) Call

In most application programs, you can provide the print data set descriptors using the change (CHNG) call interface. The IMS Spool API uses this existing technology.

The CHNG call supports two optional parameters for IMS Spool API:

- The options list parameter points to a character string containing a list of options. The IAFP keyword identifies the CHNG call as a request for IMS Spool API services.
- The feedback area parameter points to an area through which IMS can return error information to the application program when mistakes are found in the options list.

If your application creates multiple data sets with the same print options, or is wait-for-input (WFI), use the SETO call to reduce the parsing impact.

The CHNG call provides the data set characteristics in one of the following ways:

- The call can provide the data set options directly by using the PRTO option.

  When the PRTO option is used with the CHNG call, IMS activates MVS services (SJF) to verify the print options and calls MVS services for dynamic output to create the output descriptors that are used when the print data set is allocated. This is the simplest way for the application program to provide print data set characteristics. Using the PRTO option involves the most overhead, because parsing must occur for each CHNG call with the PRTO option.

  Use the PRTO option when you cannot significantly improve application performance by using pre-built text units across multiple CHNG calls.
- The call can reference a set of pre-built text units for dynamic descriptors using the TXTU option.

Application programs that can reuse text units can achieve better performance by using the TXTU option. If the application program can manage the text units necessary for dynamic output, use the TXTU option to avoid parsing for many of the print data sets.

The application program builds the text unit in the necessary format within the application area and passes these text units to IMS.

The application program can provide the print options to IMS with a SETO call and provide a work area for the construction of the text units. Using the TXTU option, IMS can perform parsing and text unit construction so that the work area passed contains the text units necessary for dynamic output after a successful SETO call. Do not relocate this work area because it contains address-sensitive information.

- The call can refer to an OUTPUT JCL statement in the dependent region's JCL by using the OUTN option.

  This is a simple way of providing print data set information if it can be used by the application program. The output JCL statements are referenced by the OUTN option in the options list on the CHNG call. When the OUTN option is used, IMS references the output JCL statement at dynamic allocation so that the IMS Spool API obtains the print data set characteristics from the output JCL statement.

The Set Options (SETO) Call

You can construct a print data set descriptor with the set options (SETO) call. The SETO call is especially useful to application programs with wait-for-input (WFI) execution. The SETO call reduces the overhead necessary to perform parsing and text unit construction of the output descriptors for a data set.

If the application program needs to reuse a set of descriptors during the scheduling of the program, the application program provides the print data set characteristics to the IMS Spool API through the SETO call. The SETO call parses the output options and builds the dynamic output text units in the work area provided by the application program. Once the application program is supplied the pre-built text units, these text units are used with a CHNG call through the TXTU parameter. They provide the print characteristics for the data set without incurring the overhead of parsing and text unit build.

It is not necessary to use the SETO option to pre-build the text units if they can be pre-built using some other programming technique.

The Output DD Statement

You can provide print data set descriptors by adding output JCL statements to the dependent region JCL. The application program passes the name of the output JCL statement in the dependent region JCL using a CHNG call with the OUTN parameter. IMS can use the output JCL statement when the print data set is dynamically allocated.

If the output statement does not exist in the dependent region's JCL, dynamic allocation fails when the first insert is done. The application program receives a status code AX indicating the insert (ISRT) failed.

Writing Data to the IMS Spool API

You can write data to the IMS Spool API using the insert (ISRT) call or the purge (PURG) call.

You can backout print data sets using the ROLL call or the ROLB call.

Do not use the SETS call or the ROLS call to backout print data sets. These calls are not supported by the IMS Spool API.

The Insert Call

The standard insert (ISRT) call writes the data to the IMS Spool API using BSAM. Because the length of the data written to the spool can be up to 32760 bytes, the BSAM write is performed directly from the application program's buffer area.

If IMS finds that the user's I/O area is above the 16 MB line, IMS moves the user's application program data to a work area below the line before performing the BSAM write. If the application area is already below the line, the write can be done directly from the I/O area. If possible, the I/O area should be below the 16 MB line. However, you do not need to take unusual steps to place the I/O area below the line unless performance shows that such action is necessary. By writing the application program's buffer without first moving it, IMS prevents problems associated with moving large blocks of data. When BSAM is used, change the format of the I/O area so that it contains the BSAM block descriptor word (BDW).

Related Reading: For more information on the format of the I/O area, see *IMS/ESA Application Programming: Database Manager*.

The PURG Call

Use the purge (PURG) call with an express alternate PCB to release a print data set to the IMS Spool API for immediate printing. When the PURG call is issued, the print data set is closed and deallocated. If the PURG call is issued against an alternate PCB that is not marked as EXPRESS=YES, no action is taken for the print data set.

You can issue the PURG call either with or without an I/O area:

- When the PURG call is issued with an I/O area, it is treated as though it were two calls. The first function handles the PURG request, and the second function inserts the data provided by the I/O area.
- When the PURG call is issued against an alternate PCB that is not marked as EXPRESS=YES, the purge function is ignored. If an I/O area is included, the data is placed in the print data set.

The PURG call differs slightly with IMS Spool API when it is used with alternate PCBs that are generated as EXPRESS=YES:

- The current print data set is closed, deallocated, and sent to JES for printing.
- Status code of A3 is returned to the application program, indicating that the alternate PCB no longer contains a valid destination.

For information on the results of the PURG call, see Table 40.

Table 40. Results of PURG Call

Type of PURG Call	Express PCB	Non-express PCB
With I/O area	The print data set is released for printing by JES. PURG call receives status code of **A3**.	The PURG call functions the same as ISRT call.
Without I/O area	The print data set is released for printing by JES. PURG call receives status code of **blanks**.	No action is taken.

ROLL and ROLB Calls

The ROLL and ROLB calls can be issued by an application program using the IMS Spool API functions. These calls result in backout (data set deletion) of any print data sets that have not been released to the IMS Spool API because of a PURG call to an express alternate PCB.

SETS, SETU, and ROLS Calls

The IMS Spool API does not support SETS, SETU, or ROLS calls. If you issue them, no action is taken by the IMS Spool API, and no special status codes are returned to show that the SETS, SETU, or ROLS call was issued by an application program. No restrictions exist on the use of SETS, SETU, and ROLS calls, because these calls can be used by the application program outside the processing of print data sets.

Special Considerations—Descriptors Allowed

The output descriptors allowed by the IMS Spool API are those supported by the `TSO OUTDES` command of the MVS system under which IMS is executing at the time that parameter validation is performed.

Related Reading: For more information on output descriptors, see *TSO EXTENSIONS Version 2 Command Reference*.

The initialization parameters you use with the IMS Spool API can have an impact on the parameters that you use to define print data set characteristics.

Controlling Print Data Sets

Application programs have some control over the IMS Spool API files IMS creates for them. This control applies to the data streams generated by the application program as well, because IMS is not sensitive to this data. You can embed control information within the data stream itself, or you can specify the IMS Spool API file options in an application program DL/I CHNG call. These processing options are specified on a data set basis and are specified on the TSO OUTDES statement.

The IMS Spool API routes print data based on output scheduling. Generally, scheduling is determined by output class (CLASS parameter), printer destination (DEST parameter), FORM, and PRMODE.

Valid Descriptors

IMS is a transparent delivery service. IMS dynamically creates IMS Spool API files based on data set processing options, writes the messages to them, and then deallocates them so that they can be processed by an IMS Spool API data set server (such as PSF).

To ensure proper disposition of print data sets, the MVS operator might need to become involved under certain conditions, such as:

- IMS emergency restart
- Dependent region abnormal termination
- Dynamic deallocation failure for a print data set

Under such conditions, the MVS operator must deal appropriately with the IMS Spool API data sets that represent in-doubt messages. IMS provides informational messages to help the MVS operator if the application program requests the proper disposition of these data sets.

Review operational procedures for unprintable data sets at your installation. Many installations have jobs that delete print data sets after a given number of days. Make sure that these jobs do not delete IMS print data sets that should be printed.

Express Alternate PCBs

The IMS Spool API supports the use of express alternate PCBs. If the application program issues a purge (PURG) call against an alternate PCB that has been generated with EXPRESS=YES, the print data set is closed and deallocated. It is available for JES processing before the termination of the IMS unit of work.

Related Reading: For more information on using the PURG call with an express alternate PCB, see "The PURG Call" on page 342.

XRF Environments

The IMS Spool API operates within XRF systems with no special support. During an XRF takeover, the IMS Spool API is affected as if a failure and normal restart occurred. If partial print data sets exist when the takeover occurs, and IMS abnormal termination processing does not execute on the primary system, the partial print data sets can be made available to IMS Spool API for printing.

Related Reading: For a description of the proper procedure for these partial print data sets, see *IMS/ESA Application Programming: Design Guide*.

Understanding Allocation Errors

The IMS Spool API interface dynamically allocates the print data set only after data is actually inserted to the data set. This reduces overhead and simplifies cleanup for abending transactions. Occasionally, errors can occur during dynamic allocation that are the results of incorrect data set print options supplied with the CHNG or SETO call. The data set print options can be parsed during the processing of the CHNG and SETO calls. Destinations can only be validated during dynamic allocation.

If any of the print options are incorrect and dynamic allocation fails when the first insert is done for the data set, the ISRT call receives a status code of AX. The IMS Spool API code provides the following to the application program:

- A status code
- An error message DFS0013E

- A diagnostic log record (67D0)

The error message shows the type of service that is activated and the return and reason codes that are responsible for the error. For example, a common failure is indicated by reason code 046C: Remote work station not defined to job entry subsystem. You see this reason code if you select an invalid destination or you select integrity option 2 (non-selectable destination) when the destination of IMSTEMP has not been defined to JES. Specifying an invalid destination in the destination name parameter of the call results in a dynamic deallocation error when IMS deallocates the print data set.

Some of the services indicated by the error message include:

DYN MVS dynamic allocation (SVC99)

OPN MVS data set open

OUT MVS dynamic output descriptors build (SVC109)

UNA MVS dynamic deallocation (SVC99)

WRT MVS BSAM write

Related Reading: If the service is for dynamic allocation or deallocation, or for dynamic output descriptor build, see *MVS/ESA System Programming Library: Application Development Guide*.

Allocation Errors

Chapter 12. Administering the Remote Site Recovery Complex

This chapter describes the remote site recovery (RSR) feature, explains how to install RSR, compares RSR to XRF, explains how to initialize RSR, and describes IMS error handling for RSR.

In this Chapter:

What Is RSR?

When your computing system is disabled, you need to recover quickly and ensure that your database information is accurate. Interruption of computer service can be either planned or unplanned. When interruption on the primary computing system occurs, you need to resume online operations with minimal delay and minimal data loss.

RSR allows you to recover quickly from an interruption of computer services at an active (primary) site. RSR supports recovery of IMS DB full-function databases, IMS DB Fast Path DEDBs, IMS TM message queues, and the IMS TM telecommunications network.

IMS database and online transaction information is continuously transmitted to a tracking (remote, or secondary) site. The tracking site is continually ready to take over the work of the active site in the event of service interruption at the active site.

Because customers require that IMS has the ability to resume online operations at a remote tracking site in the event of an extended outage (either planned or unplanned) at the active site, RSR does the following:

- Provides a remote copy of the necessary IMS DB and IMS TM log records for database and message queue recovery at the tracking site.
- Reduces the time required to resume computer service to usually less than an hour.
- Lets you select and filter out the log records that are not needed to support the defined critical environment.

- Continues to operate when the active or tracking sites or the RSR transmission facility become temporarily unavailable, and provides a way to resynchronize the sites as soon as possible.
- Provides transaction consistency between the active and tracking sites.
- Supports IMS DB and IMS DBCTL. Supports full-function databases and Fast Path DEDBs. Supports both online IMS DB and DBCTL workloads, as well as batch workloads, at the active site.
- Supports data sharing at the active site.
- Coexists with XRF (see Table 41 on page 356).
- Recognizes that DBRC is operating at the active site and, separately, at the tracking site.
- Supports standard ACF/VTAM communication protocols, so that new technology is not required for data transmission.

Requirements for Using RSR

The first and most important requirement for using RSR is that you must have a remote site available. In order for the remote site to be able to take over your critical workload from the active site, the remote site must have almost everything the active site has, including hardware, software, and at least some personnel.

Before you use RSR, you need to consider RSR's effect on the following tasks:

- Installation
  - Installing IMS with RSR at two, usually physically separate, sites.
- Operations
  - Maintaining a tracking site.
  - Establishing new procedures for:
    - Database recovery
    - System recovery
    - Network switching
- Database administration
  - Considering the DASD space requirements for the tracking site.
  - Considering the database recovery activities, such as:
    - Deciding the frequency of recovery: Do you recover only in case of disruption, or continually?
    - Synchronizing remote and primary copies (such as maintaining consistency across reorganization).
    - Sending image copies to the tracking site.
  - Maintaining the remote supporting environment, such as PSBs and DBDs.
- System programming, maintenance, and tuning
  - Writing exit routines for new functions.
  - Implementing new log structure and format.
  - Maintaining the remote supporting environment (including SYSGEN and libraries).
  - Preparing for system and database recovery (IMS restart).
  - Writing new audit procedures.
- Network administration
  - Examining your current network (VTAM) configuration and assess how the workload added by RSR will affect it.

Understanding the Basic Components of RSR

To provide remote site recovery for your IMS system, you need a primary IMS site and a remote IMS site. The primary site is where your IMS work is performed. The remote site performs no IMS work, but provides remote recovery support and is ready to perform your IMS work at any time.

This section describes the basic components and terminology of IMS remote site recovery.

Subsystems

Active IMS subsystems and XRF alternate IMS subsystems comprise the active site. A tracking subsystem comprises the remote tracking site.

An *active subsystem* is the subsystem used to perform your day-to-day work. It can consist of a DB/DC, DBCTL, or DCCTL system, as well as batch DL/I and other batch environments. The active subsystem supports both databases and terminals. Databases residing on an active subsystem are called *master databases*.

An *alternate subsystem* is an online subsystem located at the same site as the active subsystem. It acts as the XRF alternate for the active subsystem, and is therefore used only if you are using XRF at the primary site. The alternate subsystem uses the master databases of the active subsystem.

The *tracking subsystem* is usually located at a different site from the active and alternate subsystems. This is the subsystem that tracks the active site activity by maintaining backup copies of the master databases from the active subsystem. These backup databases are called *shadow databases*.

The tracking subsystem can only be an online IMS. Batch DL/I and other batch environments are not supported as tracking subsystems.

The Transport Manager Subsystem

The transport manager subsystem (TMS) provides communication services to components of RSR, using VTAM's advanced program-to-program communication (APPC) support. The transport manager:

- Dynamically allocates APPLIDs to IMS subsystems, as needed.
- Provides directory support, allowing RSR components to run on any CPC in your installation without system definition changes. This directory support also allows RSR IMS subsystems to dynamically locate IMS subsystems operating in particular roles (for example, as active or tracking).
- Provides full-duplex conversations. These conversations do not have a send or receive state, so each end of the conversation can send or receive at any time and can do so simultaneously.
- Provides a single service address space for the isolated log sender per CPC.
- Provides a simple set of interfaces for RSR components.

Related Reading: For more information on the isolated log sender, see "Isolated Log Sender" on page 351.

The RSR transport manager provides an interface similar to APPC, and includes capabilities that RSR requires when APPC is inappropriate. Because a transport manager conversation is full duplex (using two APPC conversations), it eliminates

much of the state management complexity of APPC and provides better performance in areas such as bandwidth and CPC usage.

The Log Router

The log router component of the tracking subsystem receives data from the active subsystems, stores the log data in tracking log data sets, and routes log records to individual tracking subcomponents, called *trackers*. The log router is unique to tracking subsystems; it is not found in active subsystems.

Note: In the rest of this section, the term “SLDS” (system log data set) refers to the active-IMS-generated SLDS stored and managed by the tracking subsystem at the tracking site.

The log router manages the tracking end of communication between the active and tracking sites. It initiates and accepts conversations with the active subsystem logger. These conversations enable the log router to receive log data and to communicate system control information from the active subsystem logger to the log router component.

The log router receives log data from the active subsystem loggers and writes the log data to SLDS data sets. When the log router creates an SLDS, it uses data set allocation parameters found in the IMS PROCLIB member DFSRSRxx. If more than one active logger sends log data to the tracking subsystem, data reception and log data writing proceeds in parallel. Later, you can copy the SLDS to other data sets (such as tape), and delete the original SLDS; or you can have the log router perform automatic SLDS archiving.

When the tracking subsystem operates at the database readiness level, log records are presented to the database trackers (DL/I and Fast Path) after they are written to an SLDS. If any active subsystems participate in block-level data sharing, the log records from the sharing subsystems are merged in time stamp sequence before being presented to the database trackers.

Related Reading: For more information on database readiness levels, see “Determining the Extent of Recovery” on page 354.

When an active subsystem is unable to send log data to the tracking subsystem (due to a repairable cause, such as network communication disruption) and the log router recognizes a gap in received log data, the log router contacts the isolated log sender at the active site and requests the missing log data. After the data is received, it is written to the SLDS and presented to the trackers. Until the log gap is filled, no log records beyond the gap are sent to the trackers. The records after the gap are written to the SLDS. Later, these records will be presented to the trackers in order to bring them up to date with the latest possible active subsystem log data. This procedure is called *catch-up processing*.

A stopped shadow database or area can be restarted at the tracking site, at which time the database or area will be made to match the active site with online forward recovery. *Online forward recovery* (OFR) is the process of bringing data in a database or area to a current state. The log router reads log records from previously written SLDSs and presents them to the trackers. Eventually, the log read process for OFR catches up with the current log routing and all further database updates are handled by normal tracking.

The log router records (on the tracking system log) its location in each SLDS and records the log records that have most recently been routed to the trackers. This information is used when the tracking subsystem restarts and enables log records to be routed to the trackers in the correct order.

Isolated Log Sender

Certain conditions exist that prevent IMS log data from being sent to the tracking site at the time it is written to the active subsystem's logs. These conditions might be VTAM link failures, a temporary outage of the tracking subsystem, or other repairable problems.

When the active subsystem is unable to send its log data to the tracker, but is still logging database changes, the tracker recognizes that a *gap* exists in the log. When the tracker is able, it requests the missing log data—the gap—to be sent from the active subsystem. It is the isolated log sender (ILS) that sends the missing log data to the tracker.

The ILS is functionally active only at the active site, but should also be available at the tracking site so that the ILS can be activated after remote takeover. Only one ILS is functionally active at a time for any global service group sending missing log data to the tracker, but standby ILS instances can also be started. The ILS can be started during transport manager initialization or by an operator command.

Because it is possible for a log to have a gap that spans several log data sets, the ILS can send the log data sets in parallel in concurrent conversations with the tracking subsystem.

Until the ILS fills the gap for the tracker, all subsequent log data is simply written to the tracker's log data set and is not used by the database trackers. This unused log data is *marooned* until the gap is filled. See Figure 58.

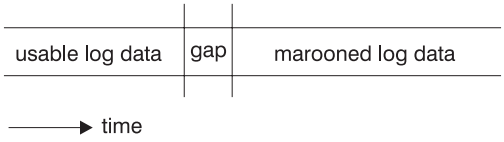


Figure 58. Marooned Log Data

DL/I Database Tracker

The DL/I database tracker receives the active subsystem's log data from the log router and the tracker updates shadow databases at the tracking site. This tracker is active only when you choose database readiness level (see "Determining the Extent of Recovery" on page 354) and LSO=S is specified in the EXEC parameters.

The tracker requires exclusive authorization for using the database before updates can be applied. It also checks with DBRC to determine if the updates should be applied. The IRLM is not required at the tracking site for database tracking.

The log router calls the database tracker periodically, to ensure a short restart time for the tracking subsystem and to record a pointer into the logs being processed to allow correct reprocessing from a particular point in the logs. These points are called *milestones.* Milestones occur periodically through the life of a tracking subsystem, and are triggered at regular intervals. A new milestone does not begin

until the previous one completes. When the log router calls the database tracker, shadow database updates are written to DASD.

Database tracking also includes an online forward recovery function. This allows catch-up database processing so that databases that were unavailable for some reason can be made current. The database tracker also uses online forward recovery to update databases when an initial or current image copy is received.

Online forward recovery is only available on tracking subsystems running at database readiness level (DLT) for databases defined as DBTRACK (database level shadowing).

Fast Path Database Tracker

The Fast Path database tracker receives the active system's Fast Path log data from the log router. This tracker is active only when you choose database readiness level (DLT; see "Determining the Extent of Recovery" on page 354). The IRLM is not required at the tracking site for database tracking.

The FP database tracker supports all the availability features of DEDBs for shadow databases: database partitioning (areas), multiple copies (MADSs), and records deactivation (EQEs). The number of area data sets at the tracker can be different than that at the active site. You must register each area you want tracked in RECON as database-level shadowing.

MSDBs are not supported by the Fast Path database tracker. VSO DEDBs are handled in the same way as non-VSO DEDBs.

As Fast Path database updates are received by the Fast Path database tracker, they are stored in an MVS data space to reduce physical I/O. The database updates are kept in the data space until either a data space threshold value is reached or the tracker decides to write the data to disk. If several updates apply to the same CI, they are accumulated and written to DASD at the same time.

DEDB update log records are held until a commit log record appears. This is done because Fast Path does not log `UNDO` log records (which DL/I uses to be able to back out changes).

Naming Conventions

In order to uniquely identify the various parts of an RSR complex, an RSR name is qualified according to a naming convention. As part of this convention, various parts of the RSR complex are subdivided and named, so that each unique part of the RSR complex is given a unique name.

A fully qualified RSR name would look like this:

```
GSGname.SGname.SYSTEMname.INSTANCEname.COMPONENTname
```

The parts that make up this name are described below.

Global Service Group

A *global service group* (GSG) is the collection of all IMS subsystems that access a particular set of databases. A global service group can span several MVS subsystems at more than one geographical location.

You can have more than one global service group. IMS systems that do not share resources are generally in separate GSGs, but are not required to be. Systems belonging to different GSGs can be connected using MSC or ISC links and can cooperate by transaction routing.

Service Group

A *service group* (SG) is a collection of all IMS subsystems that access a particular version of a GSG's resources, including the recovery control (RECON) data set. A service group usually includes one or more subsystems at a single site, with the databases and the RECON data set shared between the subsystems.

An RSR GSG is made up of two service groups: the active service group and the tracking service group. A service group name is usually the site name. When a remote takeover is required, it occurs on a service-group level: All activity of one service group is taken over by another service group.

Figure 59 shows two service groups in a very simple RSR complex.

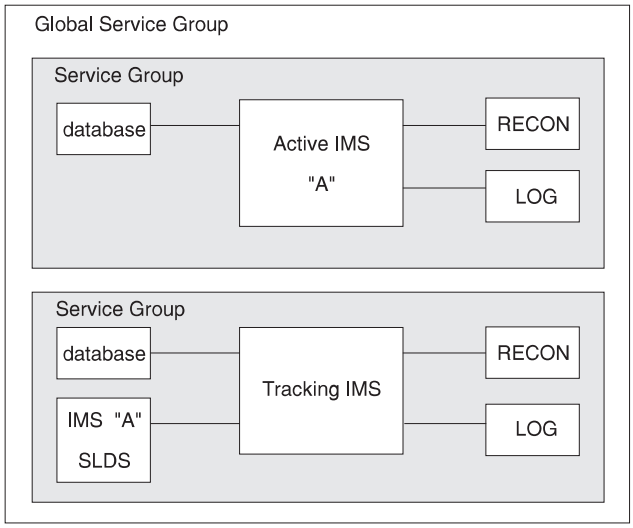


Figure 59. The Basic Elements of an RSR Complex

System

For RSR, a "system" is an MVS implementation that has one instance of the RSR transport manager running on it. Because of the need for operational flexibility, the system name can be generated dynamically.

Instance

For RSR, an "instance" is a particular, unique IMS subsystem within the MVS system. An example of an instance could be a particular DBCTL subsystem, a particular DB/DC subsystem, a batch DL/I job, or a batch utility job. For DB/DC, DBCTL, and DCCTL subsystems, the IMSID is the instance name. Several IMS subsystems running on one MVS system have the same system name but different instance names.

Component

For RSR, a "component" is a part of RSR within a particular IMS instance. Examples of components are the logger, log router, and isolated log sender.

Remote Takeover

Because “takeover” is such an integral part of RSR, this section defines what is meant by the term in the RSR context. Using RSR, you have the capability to transfer an installation’s IMS processing capacity to a remote location from an active site if the need arises. This transfer of processing responsibility is called a remote *takeover*. RSR provides two types of takeovers: planned and unplanned.

A planned takeover, initiated by the user, is an orderly transfer of operation to the remote site after a successful shutdown at the active site.

An unplanned takeover is the switching to a remote site as a result of an unexpected outage at the active location. This takeover, also user-initiated, does not require a successful active shutdown. For both takeover types, however, a successful shutdown of the tracking system is required. This shutdown is initiated by the takeover command. The user then starts the active systems at the remote location.

RSR Processing

IMS recovery processing, as well as user tasks and procedures, are unchanged from that of previous IMS releases, except that recovery is extended to the tracking site.

- The IMS log continues to be one of the basic elements in database and subsystem recovery. System log data is archived when it is sent from the active site to the tracking site. This data can be processed as it is received to keep copies of databases updated.
- Database Recovery Control (DBRC), required for subsystems using remote recovery, continues to perform central recovery control functions.

  DBRC provides the following functions for RSR:

  – Commands for defining, updating, and displaying RSR status
  – Services for the active subsystem to determine what tracking site is used, and which databases are associated with that site
  – Facilities for the tracking site to record information about log data sent from the active site and received at the tracking site
  – Services to assist in takeover
- You can use emergency restart to ensure consistency of databases and to recover resources such as message queues and scratch pads.

Determining the Extent of Recovery

After you define the active and tracking sites, you must decide what level of readiness you need. RSR provides two levels of readiness for recovery: recovery readiness and database readiness.

You control which readiness level you want in two ways:

- Use the IMS RLT (recovery level tracking) or DLT (database level tracking) startup options to determine the tracking system’s readiness level.

  RLT is a valid readiness level for any type of tracking IMS system: DBCTL, DB/DC, and DCCTL.

  DLT is not allowed for a DCCTL tracking IMS system. A tracking IMS system can have a DLT readiness level only if it is DBCTL or DB/DC.

- Use the RCVTRACK (recovery readiness tracking) or DBTRACK (database readiness tracking) registrations of databases to track individual databases.

  RCVTRACK and DBTRACK define the readiness level for databases. If you define a database as either RCVTRACK or DBTRACK, that database is considered *covered*.

Recovery Level Tracking (RLT)

For recovery level tracking (RLT), you first send copies of the master databases from the active site to the tracking site. Ensure that your databases are registered with DBRC. These copies of the master databases are not updated by the tracking subsystem after they arrive at the tracking site; that is, they are not shadow databases. The log data sent from the active site to the tracking site is archived on tracked SLDSs and kept until recovery or takeover is required.

Because the databases must be forward recovered before you use them (after a takeover), you should use recovery level tracking only for those subsystems for which recovery is infrequent or which do not require quick recovery. In most cases, a remote takeover at recovery readiness level takes several hours.

Because the databases are not shadowed, you do not require the DASD resources for the remote databases, and the tracking subsystem requires fewer CPC resources to maintain the recovery readiness level for RSR. This resource savings allows the tracking site to perform other, non-RSR, tasks while maintaining recovery readiness and tracking the active subsystem.

Also, because the databases are not shadowed, it is recommended that, before a planned remote takeover, you change the readiness level from RLT to DLT, restart the tracking subsystem, and start the databases (or areas) so that they can be automatically recovered in parallel using online forward recovery (OFR). You can periodically recover databases offline to shorten the recovery time after a remote takeover.

For an unplanned remote takeover, you must use the database recovery utility to recover the covered databases before starting the new active subsystem.

Database Level Tracking (DLT)

For database readiness level (DLT), you first send copies of the master databases from the active site to the tracking site. After the database copies are at the tracking site, the user must NOTIFY.IC to register the IC in the tracking site RECON data set. Then, the user can either do a `GENJCL.RECEIVE` to install the shadow database, or install the shadow database using some other user installation method and issuing a `NOTIFY.RECOV` command to indicate that the database is installed. When changes are made to the databases at the active site, those same changes are applied to the shadow databases. In this way, the shadow databases are always kept current, but because the tracking site's process is asynchronous with the active site's, when a takeover occurs the shadow databases might not be absolutely current.

Because the databases from the active site are shadowed by the tracker, takeover can usually occur in less than an hour.

Also because the databases are shadowed, you need to allocate permanent DASD resources for the databases being tracked.

Determining Extent of Recovery

You can choose to track only your most critical databases at the database (DBTRACK) readiness level and track your less critical databases at the recovery (RCVTRACK) readiness level. This allows you to user fewer DASD and CPC resources at the tracking site. You can change a database's tracking readiness level without having to restart either the active or the tracking subsystem, but you must issue a database recovery (/DBR) command for the database, and you must change the database's readiness level at both sites.

You can change the readiness level of the tracking subsystem (from DLT to RLT or from RLT to DLT) at any time, but when you do, you must restart the tracking subsystem. The active subsystem can be operational while you change the readiness level of the tracking subsystem.

XRF and RSR

If you consider the recovery support offered by XRF and by RSR, you might conclude that RSR function is a superset of XRF function. However, important differences exist between the kind of recovery support provided by XRF and that provided by RSR. The main difference is the goal of each: XRF is intended to facilitate recovery in cases of brief or repairable failure of the active subsystem, whereas RSR is intended to facilitate recovery in cases of lengthy or catastrophic failure of the active subsystem. Several other important differences are listed in Table 41.

Table 41. Comparison of Recovery Functions Provided by XRF and RSR

XRF	RSR
XRF relies on the same physical databases and many of the same physical resources as the active subsystem, thus yielding a single point of failure.	RSR duplicates all physical resources without a distance limitation, thus no single point of failure exists, and so an area-wide physical problem can be overcome.
XRF supports IMS DB/DC and DCCTL configurations.	RSR supports IMS DB/DC, IMS DBCTL, IMS DCCTL, and batch DL/I configurations.
XRF performs takeover on a subsystem-by-subsystem basis.	RSR remote takeover includes all subsystems that share databases.
XRF has no exposures to marooned data.	• For planned takeovers, RSR has no exposures to marooned data. • For unplanned takeovers, RSR users have exposures to marooned data.
Switching to the alternate subsystem and back again is relatively easy.	• For planned takeovers, switching back to the original site is more complex than for XRF takeovers. • For unplanned takeovers, switching back to the original site is very difficult, and requires a planned takeover (to the original site).
XRF requires some subsystems management.	RSR requires more subsystems management because of the need to replicate descriptors, programs, and other resources and objects at a second site.
XRF switch is fast, on the order of one minute.	RSR takeover is slower, on the order of one hour.

You can use RSR and XRF together or separately. After completing an RSR takeover, you can start an XRF alternate subsystem to support the new active subsystem (at the remote site). No XRF-alternate support exists for a subsystem running as a tracking subsystem.

Conversation is established prior to XRF takeover to avoid delay in sending log data after the XRF takeover. If any log data is missing after the XRF takeover, the tracking subsystem obtains it.

Defining an XRF/RSR Environment

If your installation already has an XRF implementation, defining the environment to include RSR is not difficult. Defining TMS APPLIDs in an XRF complex is similar to defining them in a multiple-CPC data sharing environment. The TMS startup commands differ from site to site only by their APPLIDs.

See Figure 60 for an example of defining RSR in your environment.

```
                  Active Site:

Active System:           XRF Alternate System:

SET APPLID(TMSA)...        SET APPLID(TMSX)...
DEFINE SYSTEM(TMST,TMSX)       DEFINE SYSTEM(TMSA,TMST)
START TMS                      START TMS
START SYSTEM(ALL)              START SYSTEM(ALL)

                  RSR Tracking Site:

Tracking System:

SET APPLID(TMST)...
DEFINE SYSTEM(TMSA,TMSX)
START TMS
START SYSTEM(ALL)
```

Figure 60. Example of Defining an XRF/RSR Environment

Be aware that setting a different instance name for each TMS requires the active and alternate systems to use different DFSRSRxx members, where TMINAME(xxxx) refers to the appropriate instance name. If online change MODSTAT tracking is used, both RSR members must have the same TRKMODS definitions. It is recommended that you make all TMSs have the same instance name to allow the active and alternate systems to use the same DFSRSRxx member. (Defining different instance names is only useful when multiple TMSs are running on the same CPC.)

The tracking subsystem uses the SSIDs and RSENAME for tracking the XRF active and alternate subsystems. When an XRF takeover occurs, the tracking subsystem is aware that the log data that the new active subsystem sends is a continuation of the log data being received from the old active subsystem.

For ILS in a multiple-CPC environment, you can choose to start ILS on any one or any number of CPCs:

Starting ILS only on the alternate system's CPC

Because the workload is generally less on the alternate system's CPC, starting ILS only on the alternate system's CPC can be beneficial. However,

doing so creates a situation in which, during an XRF takeover, both ILS and takeover processing must occur simultaneously.

Starting ILS on multiple CPCs

If the ILS that is engaged in conversation with the tracking site fails, the tracking site can obtain a conversation with another ILS.

Starting ILS on less than all CPCs.

Choosing the CPCs on which to start ILS depends on your installation's requirements.

Data Sharing and RSR

RSR supports data sharing for IMS databases. All active subsystems that share data send log data to a single tracking subsystem. The tracking subsystem is responsible for tracking all database activity for all of the active subsystems in the service group.

When the tracking site receives log data, it records it on SLDS and notifies DBRC of the data's existence. If the tracking site is operating at the database readiness level, log data from all active subsystems is passed to the database trackers. RSR ensures that data integrity is maintained at the tracking site for logs received both from active subsystems participating in block-level data sharing and from non-data-sharing active subsystems.

When a remote takeover occurs, all subsystems in the active service group are affected. If you initiate a remote takeover because one IMS system in a data sharing environment fails, the workload of the entire service group is transferred to the remote site. Therefore, using RSR to provide XRF-like coverage for isolated disruptions is probably inappropriate.

Figure 61 shows an RSR data-sharing environment.

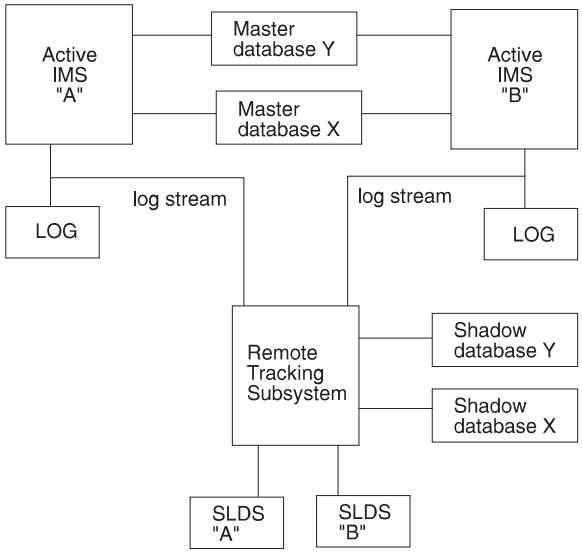


Figure 61. Data Sharing in an RSR Complex. The log data for the active subsystems is sent to the tracking subsystem.

The tracking subsystem manages data sharing so that it:

- Avoids physical sharing of databases at the tracking subsystem, thus avoiding the complexity and cost of locking and data sharing at the tracking site
- Logically merges all log records from the data sharing subsystems, thus preventing transaction and database application inconsistency
- Uses a single tracking subsystem to support many active subsystems

RSR Log Management

This section describes RSR log management for the active and tracking subsystems.

Active Subsystem

The active subsystem operates much as it did with previous IMS versions, using a control region, DL/I separate address space (SAS) region, DBRC region, dependent regions, and optionally an IRLM region. RSR also adds a new subsystem for the transport manager.

The IMS logger establishes a conversation with the log router component of the tracking subsystem using the transport manager of the active subsystem. As log buffers fill, they are sent to the tracking subsystem, prior to OLDS I/O's being initiated. When it is at the tracking subsystem, the data is not routed until the tracking subsystem confirms that the data has been written to DASD at the active subsystem.

When disruptions occur and log data is produced at the active subsystem but not sent to the tracking subsystem, you must request an additional active site component, the isolated log sender, from the tracking subsystem. This call requests all log data created during the disruption. The isolated log sender obtains data set information from DBRC and then reads and sends the data to the tracker's log router.

Tracking Subsystem

The tracking subsystem operates rather differently than the active subsystem. The tracker uses a control region, DBRC region, and a region for the transport manager. The tracker also uses a DL/I SAS region unless you specify `TRACK=RLT` or `LSO=Y`, both of which tell the tracker not to use a DL/I SAS region; this can save you the additional CPC and storage resource of maintaining the DL/I SAS region. The DL/I SAS region is only required for DL/I databases; it is not required for Fast-Path-only systems.

The log router component runs in the control region, acting as the source of log data for other components. The log router establishes a conversation with the ILS and the loggers of all operational online active subsystems in the active service group.

As log data is received from active subsystems, the log router writes the data on an SLDS. This SLDS is different than the usual IMS SLDS in that it is not an archive of an OLDS nor is it a log from a batch job; it is unique to the tracker in an RSR complex. The tracker informs DBRC of SLDS data set creation and all the log-data-set-to-database relationships. SLDSs are recorded in PRILOG and PRISLDS records in the tracking RECON data set.

The tracking subsystem uses its own log data sets to record information required for tracking subsystem restart and recovery, such as information about SLDS data

set names and current positions in the data sets. DBRC tracks the subsystem log in the RECON data set in the same way as it does in the active subsystem. Tracking subsystem log data sets are recorded in PRIOLDS and PRITSLDS records in the tracking RECON data set.

After a network or tracking subsystem disruption, gaps undoubtedly exist in the log data. After such a disruption, the log router reestablishes a conversation with the active subsystem logger and begins receiving log data again. When the log router detects the gap in the log data, it sends a request to the isolated log sender to send the missing log data. The log router then begins routing log data to the database trackers, obtaining the log data either from the local SLDS or the active subsystem.

Example of an RSR Complex

Figure 62 illustrates some of the concepts introduced in this chapter, and shows what an RSR complex might look like.

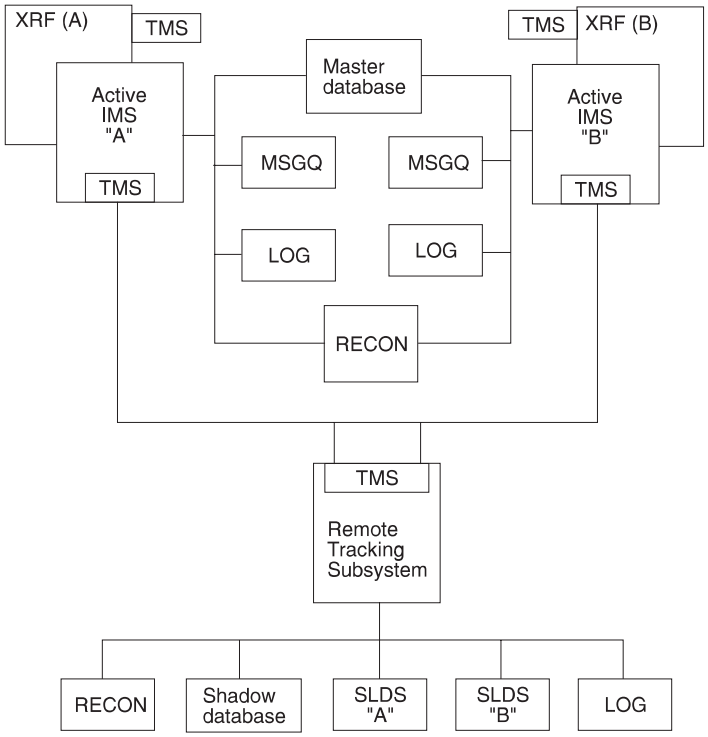


Figure 62. Example of an RSR Complex. The remote tracking subsystem tracks databases for both of the active IMS subsystems, A and B.

The IMS logs are sent from the active subsystems as they are created. This RSR complex uses XRF and data sharing. The active site has two active subsystems, A and B, which are configured with XRF and data sharing. One tracking subsystem is used for storing log data received from the active subsystems.

If one of the active site subsystems is disrupted, you should use its XRF alternate subsystem to take over its processing. If both subsystems at the active site are disrupted, this setup is designed on the assumption that you are willing to recover both subsystems, A and B, on the tracking subsystem.

When updates are made to an active subsystem, this subsystem writes log data to an online log data set (OLDS), or, for batch, to a system log data set (SLDS). At the

same time as the disk write, the active subsystem also sends this log data to the tracking subsystem. The tracking subsystem stores this data in SLDS.

At the tracking site, DBRC maintains log and database recovery information for the tracking subsystem. The tracker's RECON data set is not a mirror of the RECON data set at the active site. DBRC records log data received from both active subsystems and maintains database recovery information for the tracker in the RECON data set of the tracking subsystem.

RSR does not support active application processing against the shadow databases until a remote takeover of active processing occurs. At that point, the tracking site becomes the new active site and you can restart the active subsystems at that site. If the active subsystem participates in data sharing, you need to switch all sharing subsystems (the entire service group) to the tracking site on an RSR takeover.

General Functions

Remote site recovery at the active site provides the following general functions:

- Allocation, control and termination of conversations between active subsystem loggers or the isolated log sender and the tracking subsystem log router
- Sending of log data to the tracking site; previous missing log data is sent independently of current log data
- Support for the transition of an active site to assume the role of the tracking site after a remote takeover

Remote site recovery at the tracking site provides the following general functions:

- Allocation, control, and termination of conversations between tracking subsystem log router and the active subsystem loggers and the isolated log sender; any errors in log data transport are recognized and corrected when possible
- Receipt of log data and writing it to tracked SLDSs
- Collection of database change log data
- Application of changed data to shadow databases, if necessary
- Maintenance of the MODSTAT data sets to indicate current ACB, FORMAT, MATRIX, and MODBLKS libraries in use
- Maintenance of the tracking site's RECON data set, including status of databases and tracking and shadow logs, as well as active site information and tracking site information
- Support for tracking subsystem operations, including starting, tracking, terminating, restarting, and abnormally terminating

  Note: The tracking subsystem does not support transaction processing or the running of dependent IMS regions while in tracking mode.
- Support for a subset of IMS utilities at the tracking site while in tracking mode
- Support for the transition from tracking site to "new active" site after remote takeover

Table 42 on page 362 shows which types of IMS subsystems can be used together in an RSR environment. A DB/DC tracking subsystem can track all three types of IMS subsystem: DB/DC (IMS region type), DBCTL (DBC region type), and DCCTL (DCC region type). A DBCTL tracking subsystem can track DB/DC or DBCTL subsystems. A DCCTL tracking subsystem can only track DCCTL subsystems.

Table 42. Combinations of IMS Subsystem Types in an RSR Environment

Active Subsystem Type	DB/DC Tracker Ok?	DBCTL Tracker Ok?	DCCTL Tracker Ok?
DB/DC	Yes	Yes	No
DBCTL	Yes	Yes	No
DCCTL	Yes	No	Yes
DB Batch	Yes	Yes	No
TM Batch	Yes	No	Yes

Installing RSR

One of the most important things you must consider before you install an RSR complex is the replication of hardware and software at the two (or more) sites. Then you must consider the configuration of the active site and, more importantly, the tracking site, including the databases and data areas.

Hardware Replication

In general, hardware need not be replicated at the tracking site.

The tracking CPC must have the same architecture and hardware features as the active subsystem, unless the use of these features by IMS is optional or is controlled by specifications other than IMS system definition.

The tracking site processor should be large enough to support the expected workload after a remote takeover. However, the tracking CPC need not have the same capacity as the active processor, because a small tracking CPC can support a large active CPC if the readiness level of recovery (RLT) is used. For either recovery (RLT) or database (DLT) readiness levels, the active site CPC and the tracking site CPC can be different sizes.

Your external storage devices (DASD and tape units) need not be the same at the active and tracking sites. The tracking DASD must have the same architecture and hardware features as those at active subsystem, unless the use of these features by IMS is optional or is controlled by specifications other than IMS system definition.

If you use DASD with different track capacities than those at the active site, you should use block sizes at the tracking site that match the lowest block size at the active site, otherwise you must manage the additional differences, such as for allocation specifications. For database data sets, the VSAM CI size and the OSAM block size must be the same at both active and tracking sites.

Recommendation: If you use Fast Path and sequential dependents (SDEPs), use the same device types for your active and tracking DEDB areas. The last usable CI varies by device type, and using the same type for your DEDB areas ensures that the tracking subsystem can track SDEP inserts when you reach the end of the area.

Related Reading: For more information on SDEPs, see the *IMS/ESA Utilities Reference: Database Manager*, the *IMS/ESA Customization Guide*, and the *IMS/ESA Administration Guide: Database Manager*.

Software Replication

Because the tracking site's subsystem is effectively a duplicate of the active site's subsystem, any changes to system definitions, options, application programs, and IMS maintenance levels generally need to be made to both subsystems. You are responsible for replication of code and definitions (such as PSB and DBD definitions), and you must replicate any changes made to them.

As is the case with an XRF alternate subsystem, the RSR tracking subsystem must use the same release level of IMS (but not necessarily the same maintenance level) as does the active subsystem. You should also use the same MVS release level on each subsystem, but at a minimum, both sites must have the lowest level of MVS required by the current release of IMS running at the sites. Normally, the IMS and MVS systems should be similar enough at each site so that the output of the stage 2 of IMS system definition can be copied from one subsystem to another.

Certain requirements exist for the placement and replication of data sets for RSR. Specifically:

1. Data sets whose content must be replicated at tracking sites

   These are data sets containing definitions, procedures and code (ACBLIBx, DBDLIB, FORMATx, JOBS, MATRIXx, MODBLKSx, MODSTAT, PGMLIB, PROCLIB, PSBLIB, RESLIB, and TFORMATx). With the exception of the MODSTAT data set, you are responsible for replicating any changes to these data sets. Changes at the active site to the MODSTAT data set are tracked at the tracking site.

   If your tracking site is tracking more than one active site, it's resources are likely to be a composite of all the subsystems it tracks. In this case, it might be necessary to have separate tracking datasets for the ACBLIBx, DBDLIB, and PSBLIB datasets. You can build a proper ACBLIBx by merging the DBDLIBs into a tracking subsystem DBDLIB, merging the PSBLIBs into a tracking subsystem PSBLIB, and then running ACBGEN for the tracking subsystem. The tracking subsystem can use any FMTLIBx that contains the required formats; you do not need to make a composite FMTLIBx.
2. Data sets for which space must be allocated at the tracking site and whose content must be created specifically for the site

   These are the RECONx data sets. The contents of the RECON data sets are created by DBRC commands and by the tracking subsystem as it tracks the active subsystems.

   The RECON data set at the tracking site is neither a physical nor logical copy of the RECON data set at the active site.
3. Data sets for which space must be allocated at tracking sites

   These are all other IMS data sets not already mentioned, including the data sets that are shared by an XRF active and alternate subsystem.

   Database data sets need not have space allocated if they are not tracked or if they are not going to be maintained at the database readiness level.

Data set names do not need to be the same at the different sites. When data set names are different, these differences must be reflected in your JCL procedures, DBRC definitions, and dynamic allocation descriptors. Subsequent alterations to these procedures, definitions, and descriptors must then be made manually at each site as appropriate.

Any alterations made to MVS or to its supporting product definitions and data sets, in order to install or tune IMS, must normally be replicated at tracking sites.

DL/I Databases

In preparing for installation of an RSR complex, you need to consider the following for your DL/I databases:

- Database coverage
- Installation of shadow databases
- DBDGENs
- PSBGENs
- ACBGENs
- System definition
- Dynamic allocation

Database Coverage: RSR supports DL/I databases using the following access methods: HDAM, HIDAM, HISAM, and SHISAM. You can decide to track all databases managed by a given active subsystem or a subset of the databases. All databases that are to be tracked must be registered with DBRC.

If a database is not to be tracked (that is, not covered), RSR does nothing for that database: RSR does not maintain recovery data in the tracking site RECON data set for this database, nor does it maintain a shadow database.

If only a subset of the databases managed by an active subsystem are to be tracked, you should consider all associated database relationships, such as logical relationships or primary or secondary index relationships, and select coverage for these related databases accordingly. You also need to consider databases used by certain critical applications and select coverage for those groups of databases accordingly.

RSR does not automatically track databases that have logical relationships or primary or secondary index relationships with other databases, nor does it track databases used by an application that has all its other databases tracked; RSR only tracks those databases you ask it to track.

If you do not consider these other relationships, a database after a remote takeover might be physically available and current but be logically unavailable, because processing of the database requires the availability of related non-tracked databases.

Installation of Shadow Databases: For all those databases that are to be tracked, at either recovery (RCVTRACK) or database (DBTRACK) readiness level, you need to send an image copy of each database to the tracking site. Then you must tell DBRC about each database, using the `NOTIFY.IC` command, and register the databases, using the `INIT.DB` or `INIT.DBDS` command. The image copy is used as the base to recover the shadow database (for recovery readiness level) or as the base to begin tracking the database (for database readiness level).

For recovery readiness level tracking (RCVTRACK), you can store the database image copies on any storage medium. After a remote takeover, the databases must be forward recovered before they are ready for updates. It is recommended that you run online forward recovery (OFR) for all databases before a remote takeover.

For database readiness level tracking (DBTRACK), the image copies must be applied to the database data sets in order to be ready for updating during tracking. You install the shadow database using the `GENJCL.RECEIVE` command. DBTRACK on an RLT system is treated the same as for RCVTRACK databases.

DBDGEN: The database definitions (DBDs) at each site must match. Database data set DDNAMEs must also match, because they identify the data sets.

PSBGEN: The program specification blocks (PSBs) at each site must match. PSBs are not needed at the tracking site until immediately before restarting the tracking subsystem as the new active subsystem.

ACBGEN: The application control block generations (ACBGENs) at each site must be coordinated so that the members are kept identical. `ACBGEN` can be run at the tracking site (rather than copying the active subsystem's ACBLIB), but you must supply matching DBD and PSB input. If the tracking subsystem is tracking more than one active subsystem, you must perform a composite ACBGEN.

System Definition: System definition for a database readiness level tracking subsystem must include all databases that are to be tracked.

If only one active subsystem is to be tracked, that active subsystem's system definition can be used for the tracking subsystem. If more than one active subsystem is to be tracked and neither active subsystem's definition includes all the databases to be tracked, you must perform a special tracking subsystem system definition.

Dynamic Allocation: Database data set names need not be the same at the active and tracking sites. You can choose to assign different data set names for asset identification and control purposes.

Fast Path Databases

In preparing for installation of an RSR complex, you need to consider the following for your Fast Path databases:

- Database and area coverage
- MADSs
- Installing the databases
- MSDBs

Database Coverage: RSR supports DEDB areas, including VSO DEDBs; MSDBs are not supported. You can decide to track all databases and areas managed by a given active subsystem or a subset of the databases and areas. All databases and areas that are to be tracked must be registered with DBRC.

MADS: RSR supports multiple area data sets (MADSs) for the shadow copies of the database areas. The number of area data sets (ADSs) for each area at the remote site can be different from the number at the active site. Because no dependent regions are in the tracking subsystem, the ADS Create utility and ADS Compare utility cannot be used at the tracking site. ADS DSNAMEs and DDNAMEs need not be the same at the active and tracking sites.

You should create additional ADSs by copying (for example, using the MVS Access Methods Services `REPRO` command) the ADS just installed, and then make the new ones available using the `CHANGE.ADS AVAIL` command.

Installing Shadow Databases and Areas: For all those areas that are to be tracked, at either recovery (RCVTRACK) or database (DBTRACK) readiness level, you need to send an image copy of each area to the tracking site. Ensure that your databases and areas are registered in RECON. Send image copies to the tracking site and register these copies in RECON. The image copy is used as the base to

recover the shadow database (for recovery readiness level) or as the base to begin tracking the database (for database readiness level).

For each covered area, when you format the area (at the active site), you must include the GSG name.

For recovery readiness level tracking (RLT), you can store the area image copies on any storage medium. After a remote takeover, the areas must be forward recovered before they are ready for updates. For database readiness level tracking (DLT), the image copies must be applied to the database in order to be ready for updating during tracking, thus eliminating the need to forward recover the areas after a remote takeover. You install the image copy using the `GENJCL.RECEIVE` command.

MSDBs: Because MSDBs are not supported at the tracking site, MSDB-specific data sets, including MSDBCPn, MSDBDUMP, and MSDBINIT, are not required. If you send them to the tracking site, these data sets are not tracked.

Online Change

One MODSTAT data set must be defined for the tracking subsystem, and one additional MODSTAT data set must be defined for each MODSTAT data set that is to be tracked. Thus, tracking a non-XRF-capable IMS requires one MODSTAT data set, whereas tracking an XRF-capable IMS requires two MODSTAT data sets. But for any given active subsystem, only the MODSTAT information for the *currently active* subsystem is tracked. The dynamic allocation member (DFSMDA) for the tracking data sets is defined in the DFSRSRxx PROCLIB member at the active site.

A tracking subsystem that was previously an active subsystem is not required to use the same DD names specified for it when it was an active subsystem.

Running IMS Workload on Multiple MVS Images in an RSR Environment

In general, if you are running your IMS workload on more than one MVS image in an RSR environment, you must use MVS Global Resource Sharing (GRS) on each of the MVS images. This is because the isolated log sender (ILS) needs access to the IMS log data (OLDS or SLDS) on each of the MVS images in the RSR environment. OLDSs in this environment should have unique data set names and exist on disks shared among the MVS images.

ILS and the OLDS on IMS TM or DBCTL Subsystems

Because the isolated log sender (ILS) can access data on disk without operator intervention, it is highly desirable to provide an active IMS with a large amount of OLDS space. With lots of space, isolated log data can be accessed and sent quickly in many cases, without the need for tape mounts to read archived SLDS data sets.

Because the ILS cannot access data on an OLDS until the active subsystem is finished writing to it, each individual OLDS should not be made too large; this can delay the sending of isolated log data. IMS commands can be used to cause an OLDS switch, but this requires operator intervention.

Because the ILS is unable to read OLDS data after completion of archive, it is operationally desirable to be able to defer archive operations after network or tracking site outages (assuming an adequate number of OLDS data sets are defined to allow continued IMS operation).

If it is ever possible that the OLDS archive utility (DFSUARC0) can be executed on an MVS image different than the MVS image where the ILS is running, you must use MVS Global Resource Sharing (GRS) on the MVS images. This is because a combination of MVS ENQ/DEQ and DBRC are used to prevent exposure to the ILS reading an OLDS that is being re-used by a TM or DBCTL IMS system. After an OLDS has been archived, it is available for re-use by the TM or DBCTL system and must be considered unavailable for the ILS.

ILS and Batch Subsystems

If you allocate your batch log data sets on disk, conflicts with the ILS must be avoided. This is particularly true when the log data sets are reused or scratched, even after they are copied by the archive utility. Because of potential conflicts, the ILS allocates SLDS data sets with disposition `OLD` to prevent sharing and inadvertent scratching. In order for this to be effective when reuse or scratching can be executed on an MVS image different than the MVS image where the ILS is running, you must use MVS Global Resource Sharing (GRS) on the MVS images.

After a batch log has been copied and DBRC RECON updated with the change, DBRC provides the new data set to the ILS if the log data is required. Situations can exist where the previous name is given to the ILS and that data set is not yet sent. If this is the case, you can use the `DISPLAY ILS` command to check for pending data set transfers.

Initializing RSR

This section describes how to initialize RSR:

- “Initializing the Active Site”
- “Initializing the Tracking Site” on page 373

Initializing the Active Site

To initialize the active site, see the following:

- “Initializing the Transport Manager Subsystem”
- “Initializing the IMS Logger” on page 368
- “Initializing DL/I Batch” on page 368
- “Defining Master and Secondary Master Terminals” on page 369
- “Configuring the System Definition for RSR” on page 369

Initializing the Transport Manager Subsystem

You can initialize the transport manager subsystem (TMS) with an MVS `START` command or by submitting a job.

You can issue transport manager subsystem commands from a SYSIN data set at component start, or from an MVS console using the MVS `MODIFY` command.

Related Reading:

- For more information on the transport manager subsystem startup commands, see *IMS/ESA Operator’s Reference*.
- For information on the other TMS commands, see *IMS/ESA Operations Guide*.

A typical command sequence for starting the transport manager subsystem is:

1. SET

   The `SET` command allows you to specify various parameters that generally stay in effect while the transport manager task is running.

2. DEFINE SYSTEM

   Use the DEFINE command to define one or more transport manager subsystems in different CPCs to support IMS components for one or more RSR global service groups.

   You can specify one or more DEFINE SYSTEM commands when starting a transport manager subsystem.

3. START TMS

   Use the START TMS command to make transport manager functions available to other address spaces.

For each command, you see error messages if command parsing errors or command sequence errors occur. Otherwise, you see a message indicating successful completion of the command.

Figure 63 shows sample commands used to start the transport manager subsystem:

```
SET APPLID(SYSN1) APPLCOUNT(10) PASSWORD(HOHO)
DEFINE SYSTEM(SYSN2,SYSN3)
START TMS
```

Figure 63. Sample of Transport Manager Subsystem Start Commands

The START TMS command causes a VTAM access-method control block (ACB) for the transport manager subsystem to be opened and logons to be enabled. The transport manager subsystem then attempts to allocate conversations with any other transport manager subsystems it knows about (from the DEFINE SYSTEM command). Whenever conversations are allocated between transport manager subsystems, t

Dynamic definition of global service groups, service groups, and remote transport manager subsystems is supported. When an RSR component identifies itself to a TMS, its name is distributed to other transport managers, thus supporting a dynamic distributed directory. When a component issues an allocate request to a different component, the TMS supporting the requester sends the request to the transport manager subsystem that supports the requested component. If the requested instance and component are found, allocation proceeds; otherwise, it fails.

Initializing the IMS Logger

The logger obtains the service group names for the tracking service group from DBRC. It then attempts to identify to and allocate a conversation with the log router at the tracking site. If errors occur, various messages are issued, but logger initialization continues. Errors do not prevent subsequent attempts to initiate these initialization processes for IMS online subsystems.

The OLDS block size should be no larger than 32708 bytes. If the OLDS block size is larger, then the logger does not establish a conversation with the tracking subsystem, and real-time log transport is not available. The log data is sent to the tracking site at a later time by the isolated log sender.

Initializing DL/I Batch

A batch job specifies the GSG name when it signs on to DBRC. Batch is not allowed for tracking service groups; therefore, if the local service group is not the active one, the signon request is rejected. Batch jobs that do not specify a GSG name cannot update databases that have an associated GSG name; if update is attempted, the authorization fails.

During initialization, the batch logger obtains the name of the remote service group from DBRC. The logger attempts to identify to the transport manager subsystem and establish a conversation with the tracking subsystem. If the logger cannot establish conversations with any tracking subsystems, real-time log transport is not available for batch execution. The batch log data set is sent to the tracking site at a later time by the isolated log sender.

The batch log data set block size should be no larger than 32708 bytes. If this block size is larger, then the logger does not establish a conversation with the tracking subsystem.

Defining Master and Secondary Master Terminals

RSR supports the separate definition of a tracking subsystem master terminal and a secondary master terminal using the *MTOID* parameter in the DFSRSRxx procedure library member. Which DFSRSRxx member you use depends on the specification of the *RSRMBR=* parameter in the IMS (DBC or DCC) procedure.

You can override the defined VTAM application identifier for the IMS subsystem by using the *APPLIDn=* parameter in the IMS procedure.

Because the roles of active and tracking sites are reversed after a remote takeover, you should define both the active and tracking subsystems to allow each to be easily restarted as either an active or tracking subsystem.

Related Reading: For more information on these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Configuring the System Definition for RSR

RSR allows you to provide separate master terminal operators for active, XRF-alternate, and tracking subsystems at both the active and tracking sites. You define the additional tracking subsystem MTO and secondary master by specifying *MTOID=3* in the IMS (DBC or DCC) procedure, and by specifying the three names in the COMM and TERMINAL macros. You can also use the *APPLIDn=* override for the IMS `/START` command.

Because no third *ADDR=* parameter is supported in the TERMINAL macro, BTAM master and secondary master terminals for RSR are not supported.

The following parameters are very important to the proper configuration of an RSR DC environment:

- APPLID=(*name1,name2,name3*) (in the COMM macro)

  name3 is used as the APPLID of the RSR tracker. *name3* can match either *name1* or *name2*, but no two APPLIDs can be active in the network simultaneously.
- APPLID1=*name1*,APPLID2=*name2*,APPLID3=*name3* (in the IMS procedure)

  These can be specified in the startup parameters to override the APPLID names in the COMM macro. The requirement that the APPLIDs be unique makes their specification in the startup parameters highly recommended, especially if you want to run with common system definitions or common procedures.

  Because APPLID1, APPLID2, and APPLID3 are specified in the COMM macro, they do not apply to a DBCTL tracking subsystem.
- USERVAR=*uname* (in the IMS procedure)

  This parameter is specified in the startup parameters and is required if RSR is specified or if IMS is RSR capable. Furthermore, single IMSs (that is, not part of

either an XRF or an RSR complex) must specify the USERVAR parameter. However, tracking subsystems do not normally require the use of the USERVAR parameter.

For an XRF active system, this parameter overrides the XRF USERVAR specified in the IMS procedure library member DFSHSBxx.

- MTOID=*n* (in the IMS procedure)

  This parameter must be part of the tracking subsystem's procedure. It refers to which MTO name is to be selected from the TERMINAL macro's NAME list for the tracker. This number (1, 2, or 3) also indicates which password is to be selected from the COMM macro's PSSWD list when opening the VTAM ACB for the tracker. As many as three names can be specified in an XRF environment or an RSR tracking environment. You cannot override this parameter (as there is for APPLID) in the startup parameters. The default value is MTOID=1; only for a tracking subsystem can you specify MTOID=3.

The TERMINAL macro has three NAME= parameters for VTAM terminals. Note that *name3* is used only for a tracking subsystem. The COMM macro allows you to specify a third APPLID and password for the tracking subsystem.

The following table, Table 43, shows the possible combinations of system definition and procedure parameters for RSR.

Table 43. Relationship between IMS DC System Definition and IMS Procedures for RSR

Terminal parameter	**HSBID=1**	**HSBID=2**	**MTOID=3**
APPLID	Application Name1	Application Name2	Set by APPLID3 or system definition
PASSWD	Password1	Password2	Password3
NAME (for master terminal)	Name1	Name2	Name3
NAME (for secondary master terminal)	Name1	Name2	Name3

Notes:

- For the master terminal, Name1 cannot be the same as Name2. The same is true for the secondary master terminal.
- Name3 for both the master terminal and the secondary master terminal can match the MTO names in the HSBID1 or HSBID2 positional parameter of NAME. In this case, all master terminals involved must be defined as VTAM local terminals.
- You need to specify MTOID=3 for the tracking MTO when you have an active and alternate XRF subsystem at the active site, you want a different terminal for the tracking subsystem, and you want to use the same system definition that the active subsystem is using. This parameter is only valid when used for an RSR tracking subsystem.

See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for more information.

The following figure shows a possible configuration for a single IMS instance with XRF, RSR, and XRF after remote takeover. In the figure, it is assumed that separate master and secondary master terminals exist for each of the four MVS subsystems that the IMS active, alternate, or tracking subsystems can run on. At any given time, however, only one active, one alternate, and one tracking subsystem is started.

Figure 64 on page 371 shows a master terminal configuration.

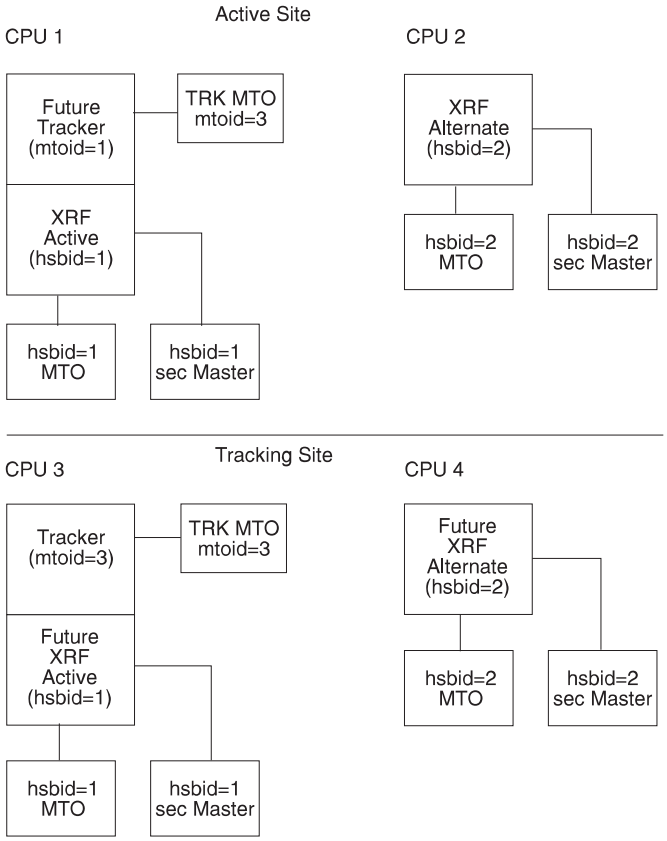


Figure 64. Master Terminal Configuration. RSR supports up to four separate Master and Secondary Master Terminals from a single system definition.

Table 44 presents another way of viewing the system definition and procedure parameters and how they interrelate.

The parameters and their sources are:

Parameter	Source
MTOID=n	RSR Proc
PSSWD=(a,b,c)	COMM macro
APPLID=(a,b,c)	COMM macro
APPLID1,2,3	Startup parameters
USERVAR=a	Startup parameters

Table 44. IMS, XRF, and RSR Configurations. How They Relate to Various System Definition and Procedure Parameters

Parameter	Basic IMS	Basic IMS with XRF	Basic IMS with RSR	IMS with RSR; XRF at Active Site	IMS with RSR; XRF at Remote Site	IMS with RSR; XRF at Both Sites
MTOID=n	N/A	N/A	Y	Y	Y	Y
PSSWD=(a,b,c)	a	a,b	a,b,c	a,b,c	a,b,c	a,b,c
APPLID=(a,b,c)	a	a,b	a,b,c	a,b,c	a,b,c	a,b,c

Table 44. IMS, XRF, and RSR Configurations (continued). How They Relate to Various System Definition and Procedure Parameters

Parameter	Basic IMS	Basic IMS with XRF	Basic IMS with RSR	IMS with RSR; XRF at Active Site	IMS with RSR; XRF at Remote Site	IMS with RSR; XRF at Both Sites
APPLID1=, APPLID2=, APPLID3=	N/R	N/R	a	a,b	a,b	a,b
USERVAR=a	N/A	N/A	Y	Y	Y	N/A
XRF USERVAR	N/A	N/R	N/R	Y	Y	N/R

Notes:

1. Symbols:

   N/A Not Applicable

   N/R Not Required

2. For the *APPLID* keyword in the startup parameters, the "c" APPLID name, for the tracker, is always optional and only serves as an override. In cases where three names are possible, all three names can be specified, and serve as overrides if specified on the startup parameters.
3. All systems are assumed to be running with VTAM terminals. The XRF configuration requires that the VTAM application name be the same as the ACB name, but it is recommended this be done in the VTAM definitions for all configurations.

The following list provides detail on each configuration used in Table 44 on page 371:

Basic IMS
: The simplest IMS system. Nothing new is required for an IMS in this configuration. However, the *APPLID* in the startup parameters allows the *APPLID* in the COMM macro to be overridden. Note that the *PSSWD* in the COMM macro cannot be overridden. *PSSWD* selection for the tracker is controlled by the *MTOID* parameter.

Basic IMS with XRF
: The simplest XRF complex. Nothing new is introduced, except the startup parameter override option for the *APPLID*s in the COMM macro. The USERVAR startup parameter is not required; if specified it overrides the XRF USERVAR.

Basic IMS with RSR
: The simplest RSR configuration. A single IMS is started when the tracking subsystem detects takeover for the single active IMS. *MTOID* is required by the tracker to select the proper MTOs from the primary and secondary MTO-names lists. *APPLID*s are required in the startup parameters, and they must be different in the IMS procedures used at the active and tracking sites. Similarly, the *USERVAR* parameters must match in the IMS procedures at the two sites.

IMS with RSR and XRF at Active Site
: In this configuration, the XRF complex exists only at the active site. A single IMS will be started at the tracking site. *MTOID* is required by the tracker to properly select the MTOs from the primary and secondary MTO-names lists. The *APPLID*s are required in the startup parameters and they must be different in the IMS procedures used at the active and tracking sites. The *USERVAR* startup parameter is required in the IMS procedure that is used to bring up the active system at the remote site after a remote takeover. This parameter need not match the *APPLID* in the startup parameters at

this site. This parameter must be specified and must match the XRF USERVAR being used at the active XRF site. The APPLIDs on the startup parameters must be carefully specified: *APPLID1* at the active site must **not** match the *APPLID1* at the tracking site. That is, the active site should specify *APPLID1=a*, and the tracking site should specify *APPLID1=b*.

IMS with RSR and XRF at Remote Site

In this configuration, the XRF complex exists only at the remote site. *MTOID* is required by the tracker to select the proper MTOs from the primary and secondary MTO-names lists. The *APPLID*s are required in the startup parameters, and they must be different in the IMS procedures used at the active and tracking sites. The *USERVAR* startup parameter is required in the IMS procedure used at the active, non-XRF, IMS site. This parameter need not match the *APPLID* in the startup parameters at this site. This parameter must be specified and must match the XRF USERVAR that will be used at the remote site after a remote takeover.. The *APPLID*s on the startup parameters must be carefully specified: *APPLID1* at the active site must **not** match the *APPLID1* at the tracking site. That is, the active site should specify *APPLID1=a*, and the tracking site should specify *APPLID1=b*.

IMS with RSR and XRF at Both Sites

In this configuration, the XRF complex exists at both the active and RSR sites. *MTOID* is required by the tracker to select the proper MTOs from the primary and secondary MTO-names lists. The *APPLID*s are required in the startup parameters and they must be different in the IMS procedures used at the active and tracking sites. The *USERVAR* parameter is not required at either site. If specified, at either site, it overrides the XRF USERVAR and must also be specified at the other site. The *APPLID=(a,b)* on the startup parameters must be carefully specified: All APPLIDs must be unique. That is, the active site should specify *APPLID=(a,b)*, and the tracking site should specify *APPLID=(c,d)*.

Initializing the Tracking Site

This section describes the activities associated with initializing the tracking site.

Setting Execution Parameters and Configuration

The *TRACK=* parameter of the IMS procedure determines whether the control region being started performs RSR tracking.

Related Reading: For more information on the *TRACK=* parameter, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Use the *RSR(NO)* parameter in the DFSRSRxx PROCLIB member of the active site to disable RSR functions. Disabling RSR function by using the *RSR(NO)* parameter is only necessary if you specify the *GSGNAME* in the system definition. The *RSR()* parameter is only meaningful for active subsystems and is ignored for a tracking subsystem.

DL/I database tracking is not initialized if you specify *LSO=Y* in the IMS procedure for the tracking subsystem. This assumes that only Fast Path database tracking is to be performed.

Initializing the Transport Manager Subsystem

The transport manager at the tracking site is initialized in the same way as at the active site.

Related Reading: For more information on initializing the transport manager at the active site, see “Initializing the Transport Manager Subsystem” on page 367.

Initializing DL/I Database Tracking

DL/I database tracking is only activated in a database level tracking subsystem; that is, a subsystem for which *TRACK=DLT* is specified in the IMS procedure. DL/I database tracking is initialized during tracking subsystem initialization. It creates and initializes the DL/I database tracking data space.

A DL/I separate address space (SAS) region is required to process log records that keep the shadow full-function databases updated. Start this region as you would in any normal IMS subsystem.

The *PST* parameter on the *EXEC PARM=* statement of the IMS procedure specifies the maximum number of PSTs used for DL/I database tracking. The *MAXREGN* number you specify during system definition is used, if it is not overridden. The number should be approximately twice the number of DL/I PSTs used in all of the active subsystems tracked by this tracking subsystem. As a minimum, 2 PSTs are used.

Initializing Fast Path Database Tracking

Before Fast Path database tracking can begin, you must:

- Specify a database readiness level tracking subsystem (specify *TRACK=DLT* in the IMS procedure).
- Specify Fast Path during system definition (include an FPCTRL macro in your system definition).

During initialization, Fast Path database tracking:

- Loads required modules
- Creates and initializes its data space

Some processes normally done during initialization (in non-RSR subsystems) are skipped to reduce storage requirements and minimize initialization time for the tracking subsystem.

Master and Secondary Master Terminal Definitions

RSR supports the separate definition of a tracking subsystem master terminal and a secondary master terminal using the *MTOID* parameter in the DFSRSRxx procedure library member. Which DFSRSRxx member you use depends on the specification of the *RSRMBR=* parameter in the IMS procedure.

The master terminal and secondary master terminal cannot be BTAM terminals at the tracking site.

You can override the defined VTAM application identifier for the IMS subsystem by using the *APPLIDn=* parameter in the IMS procedure.

The *HSBID=* parameter is ignored for a tracking subsystem; the information is obtained instead from the *MTOID* parameter in the DFSRSRxx member and from the *APPLID3* parameter in the IMS procedure. Whether or not your active subsystem uses XRF, the tracking subsystem always uses *NODE3* (from the TERMINAL macro), *PSSWD3* (from the COMM macro), and *APPLID3* (from the IMS procedure).

Because the roles of active and tracking sites are reversed after a remote takeover, you should define both the active and tracking subsystems to allow each to be easily restarted as either an active or tracking subsystem.

Related Reading: For more information on these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

System Definition and Procedure Requirements for RSR

A single active online IMS environment does not require a separate system definition for the tracking subsystem. However, additions to the combined system definition for an RSR environment are necessary. You need a separate tracking subsystem system definition in the following cases:

- The tracking subsystem is to track more than one active subsystem, and no one system definition sufficiently defines all of the databases for the tracking subsystem. You can eliminate this separate system definition requirement if you enlarge one of the active subsystems to include the additional databases.
- Batch DL/I jobs use databases that are not defined to the active subsystem.
- You want to eliminate definitions for non-tracked databases from the system definition of the tracking subsystem.
- You want to reduce the data communications definitions from the system definition of the tracking subsystem.

IMS Error Handling for RSR

This section describes IMS error handling for RSR at the active site and the tracking site.

The Active Site

The components involved in error handling for the active site include the transport manager, the isolated log sender, the online logger, and the DL/I batch logger.

Transport Manager Subsystem

The transport manager isolates IMS components from transport manager failures and from VTAM failures. The general procedure is to take an SDUMP for diagnostic purposes, recover damaged structures if possible, and retry if possible. Retry might result in a return code's being given to the IMS component invoking the function.

Failures that occur in the transport manager address space do not, in general, cause termination of IMS components. If the transport manager cannot recover from a particular failure, IMS components are informed of the failure and can continue to use the existing conversations. At this point, new conversations cannot be allocated (on the active processor) because of the failure. If new conversations must be allocated, a new transport manager must be brought up.

Do the following for all IMS active subsystems at the same site as the failed transport manager to start new transport manager conversations for these active subsystems:

1. Issue a `/STOP SERVGRP` command to break the communications with the old transport manager.
2. Issue a `/START SERVGRP` command to start communications with the new transport managers.

These steps are required if a need for the isolated log sender arises after a transport manager address space failure. Note that the current conversations

between the active subsystem and the tracking subsystem continue to run, but any new transport managers brought up do not know about these conversations.

If a transport-manager-to-transport-manager conversation fails, an immediate attempt is made to reallocate the conversation, in case an alternate path is available through the VTAM network. If that attempt fails, periodic attempts to reestablish a conversation are made based on a user-specified timer interval. VTAM errors on transport manager conversations are identified by transport manager error messages (prefix ELX) for diagnostic use.

Isolated Log Sender

In the event of allocation failures or permanent I/O errors on an active subsystem log data set, the isolated log sender attempts to use a dual copy of the log data set, if one exists. If no dual copy is available, or if the attempt to use one fails, you need to recover the log. After the log data set is recovered, you can issue a /START ISOLOG command on the tracking subsystem to reinitiate isolated log transport processing.

Related Reading: For more information on the /START ISOLOG command, see *IMS/ESA Operator's Reference*.

If log recovery cannot be performed, you must create new database image copies and transport and reinstall them at the tracking site.

Abnormal termination of an ILS instance or of the copy of DBRC in the TMS address space does not terminate the TMS. You can recover from these kinds of failures by using a START ILS command.

It is recommended that you run the ILS on at least two different MVS subsystems. Although the log router communicates with only one ILS at a time, having a second ILS available in case of failure of the first ILS allows for continuous ILS function. Having a second ILS can prevent problems associated with:

- ILS failure
- DBRC failure leading to ILS failure
- Partial network failures
- TMS, MVS, VTAM, or CPC failures

If you do have a second ILS acting as a standby, both ILSs must have access to the OLDS and SLDS data sets. In order to control access to the OLDS and SLDS across multiple MVS images, you need to use MVS global resource sharing (GRS).

Related Reading: For more information on MVS GRS, see "Running IMS Workload on Multiple MVS Images in an RSR Environment" on page 366.

Online Logger

The IMS online logger sends log data to the tracking subsystem before initiating I/O to the active subsystem OLDS. The Log Filter exit routine is given control as part of the send process (see *IMS/ESA Customization Guide*), but if a failure in the exit routine occurs, the routine abends.

Other than a VTAM overload, failures in normal operation of the send process result in termination of the conversation involved. VTAM overloading causes the online logger to suspend sending log data to the tracking subsystem, but the conversation is not terminated. This does not preclude subsequent attempts to reestablish the conversation after it has been terminated. Unless you issue a /STOP SERVGRP command, the IMS online logger attempts to resume suspended conversations and

reconnect terminated conversations at each OLDS switch. If you want a faster reconnect attempt, the MTO can issue the /START SERVGRP command.

Attempts to reestablish failed conversations can be prevented by issuing the /STOP SERVGRP command. You might want to prevent these attempts if communications to the tracking site or the tracking subsystem are expected to be lost for an extended period of time.

DL/I Batch Logger

The batch logger sends log data to the tracking subsystem before initiating I/O to the active site SLDS. The Log Filter exit routine is given control as part of the send process (see *IMS/ESA Customization Guide*), but if a failure in the exit routine occurs, the routine abends.

Failures in normal operation of the send process result in termination of the conversation involved.

The Tracking Site

The aspects involved in error handling for the tracking site include the log router, DL/I database tracking, Fast Path database tracking, online forward recovery, and online change.

Log Router

The three general classes of error conditions recognized by the log router are: system or internal errors, communication errors, and log media errors.

System Errors: System errors result from system resource shortages (for example, a shortage of storage) or are IMS internal logic errors.

Whenever possible, the effect of system errors is limited to the current “process”. For example, if the log router is unable to obtain storage for the control blocks and data areas required to manage a conversation with an active subsystem, the conversation is terminated. In this case, the active subsystem log is obtained later during isolated log transport.

Communication Errors: Communication errors affect the conversations between the log router and active site components (loggers and isolated log senders) or affect the connection between log router and the TMS.

Communication errors in a conversation result in the loss of contact with the active site component. The IMS operator is informed of the error by means of an error message. After the problem is resolved and communications are reestablished, normal processing resumes. In the case of active loggers, the current log data is received from the logger, and any missing log data is obtained by the isolated log sender. If a conversation with the isolated log sender is lost, log transport resumes at the point of error when communications are reestablished.

If the connection between the log router and the TMS is lost, processing on current conversations continues, if possible. However, no new conversations can be established until the log router is reconnected with the TMS. After the problem is resolved, you must issue a /STO SERVGRP command followed by a /STA SERVGRP command to reconnect the log router to the TMS and all active subsystems. You should wait until all batch jobs or BMPs are complete before issuing the /STO SERVGRP command.

If VTAM terminates, all conversations are lost. After VTAM and the TMS are restarted, you must issue a /STA SERVGRP command.

Log Media Errors: Log media errors result from I/O errors on the tracked SLDSs. The following types of error can occur:

- Write error on a tracking SLDS

  If single SLDS logging is used, or if an error occurs on both SLDSs, the current SLDS is closed and DBRC is notified that the SLDS is invalid. DBRC deletes the record of the invalid SLDS in the RECON data set. A message is issued identifying the SLDS, and a new SLDS is created; the log router proceeds, obtaining any missing log data.

  If dual logging is used, the SLDS in error is closed immediately and the remaining SLDS is closed as soon as previously initiated writes have completed. DBRC is informed of the invalid SLDS. You can choose to ignore the problem and continue with a single copy of the SLDS, or you can copy the good SLDS to a new data set and use the DBRC CHANGE.PRILOG (or CHANGE.SECLOG) command to inform DBRC that the SLDS is valid.

- Read error on a tracking SLDS or archived SLDS

  Read errors on tracking SLDSs can occur during catch-up processing, online forward recovery (OFR), or auto-archive.

  If only a single copy of the SLDS exists (or an error occurs on both copies), the current operation is terminated, and the record of the SLDS is deleted from the RECON data set. Any missing log data is obtained from the isolated log sender.

  When dual SLDSs exist, the second copy is opened and processing proceeds (assuming no error occurs on the second copy). As long as successive blocks can be read from one of the data sets, the operation proceeds.

  In either case, an error message is issued. The operator must restart the OFR processing, which was interrupted by the read error, if necessary.

  Related Reading: For more information on restarting the OFR processing, see *IMS/ESA Operations Guide*.

- Write error on an archive data set (SLDS or RLDS)

  Write errors during archive cause termination of the current archive operation. Eventually a new archive operation is initiated, and the log router retries the failed archive.

DL/I Database Tracking

DL/I database tracking can tolerate some unexpected return codes from other RSR components, such as from DBRC, as it tries to obtain database authorization. When it receives an unexpected error, database tracking issues a message that identifies the resource (such as the database name), and the error or unexpected return code.

Rather than bring down the entire tracking subsystem, database tracking then initiates “STOP” processing for the specific database, the smallest appropriate unit of resource associated with the error. “STOP” processing includes the termination of tracking for the database, closing the database, and unauthorizing and deallocating the database.

After an error in a specific database, tracking continues for other databases that are to be tracked.

Fast Path Database Tracking

Fast Path database tracking can tolerate some unexpected return codes from other RSR components, such as from DBRC, as it tries to obtain area authorization.

When it receives an unexpected error, Fast Path database tracking issues a message that identifies the resource (such as the area name), and the error or unexpected return code.

Rather than bring down the entire tracking subsystem, Fast Path database tracking then initiates “STOP” processing for the specific area, the smallest appropriate unit of resource associated with the error. “STOP” processing includes the termination of tracking for the area, elimination of associated log records from the data space, and unauthorizing and deallocating the area.

After an error in a specific area, tracking continues for other areas that are to be tracked.

The Fast Path database tracker handles I/O errors as follows:

- Read Errors

  Read activity is done in READ ANY mode. If the record can be read from at least one ADS, the Fast Path database tracker is not aware of the read error. If the record cannot be read from any ADS, the Fast Path database tracker stops tracking the area.

  If you have an error in one ADS, copy the other one (which has no error) immediately, then discard the one with the error.
- Write Errors

  An error queue element (EQE) is created for the control interval in error, just as is done in a non-RSR environment.

When additional ADSs need to be added to the existing area, do the following:

1. Stop the tracked area, using the `/DBR AREA areaname` command.
2. Create additional ADSs, using the Access Method Services `REPRO` command.
3. Register new ADSs with DBRC, using the `INIT.ADS` and `CHANGE.ADS AVAIL` commands.
4. Start online forward recovery for the area, using the `/START AREA areaname` command.

Note: The ADS CREATE Utility can not be used at the tracking site because the tracking site has no dependent regions.

Online Forward Recovery

When online forward recovery (OFR) for shadowed databases and areas is stopped or fails, the point at which OFR stopped on the SLDS is recorded for each database and area undergoing online forward recovery. This point becomes the “restart position” for OFR when the database or area starts again.

Online Change

I/O errors that occur when attempting to read or write to a tracking MODSTAT data set do not cause the tracking subsystem to fail. Information about the error is saved so that the read or write can be retried. The read or write is retried when the next checkpoint or online change record (X'70') is received.

Tracking MODSTAT error messages are issued when the error first occurs and when each online change record is received. Error information is saved across tracking subsystem restarts. All failed reads and writes are retried during tracking system restarts, and appropriate messages are issued.

Establishing IMS Security

This section provides information on IMS security relative to RSR.

Transport Manager Subsystem

The transport manager subsystem provides basic security by allowing you to define an ACB password to secure the access to the transport manager APPLID. You need to protect against unauthorized access to libraries with VTAM definitions by carefully using RACF or an equivalent security product.

The transport manager subsystem only communicates with a logical unit (LU) if that LUname has been specified on a `DEFINE SYSTEM` command. Combined with network security, this prevents transport manager from communicating with unauthorized programs or terminals.

All users of the transport manager subsystem must be authorized programs; that is, they must be APF-authorized or must run in system supervisor state.

IMS Terminal Security

It is important to keep terminal-related passwords synchronized between the active and tracking sites.

Related Reading: For a description on maintaining terminal-related passwords, see *IMS/ESA Administration Guide: Transaction Manager*.

Part 3. Appendixes

Bibliography

This bibliography includes all the publications cited in this book, including the publications in the IMS library.

Batch Terminal Simulator: General Information Manual, GH20-5522

DATABASE 2 Application Programming Guide, SC26-4377

DataPropagator NonRelational MVS/ESA Administration Guide, SH19-5036

DataPropagator NonRelational MVS/ESA An Introduction, GH19-5034

DataPropagator NonRelational MVS/ESA Reference, SH19-5039

MVS/DFP Access Method Services for the Integrated Catalog Facility, SC26-4562

MVS/ESA Initialization and Tuning (Ver. 4) Reference, GC28-1635, or *MVS/ESA System Programming V5 Initialization and Tuning Reference*, SC28-1452

MVS/ESA SP Version 5 Planning: Workload Management, GC28-1493

MVS/ESA SP Version 5 Programming: Sysplex Services Guide, GC28-1495

MVS/ESA SP Version 5 Programming: Sysplex Services Reference, GC28-1496

MVS/ESA System Programming Library: Application Development Guide, GC28-1852

MVS/ESA System Programming Library: Initialization and Tuning, GC28-1828

MVS/ESA System Programming Library: System Modifications, GC28-1831

MVS/ESA System Programming V4 Initialization and Tuning Guide, GC28-1634, or *MVS/ESA System Programming V5 Initialization and Tuning Guide*, SC28-1451

MVS/ESA Version 4 System Modifications, GC28-1636

Network Program Products Planning, SC30-3351

Resource Access Control Facility (RACF) Auditor's Guide, SC28-1342

Sysplex Overview, GC28-1208

System Programming Library: Resource Access Control Facility (RACF), SC28-1343

TSO EXTENSIONS Version 2 Command Reference, GC28-1881

VTAM Operation, SC23-0113

The X.25 Interface for Attaching IBM SNA Nodes to Packet Switched Data Networks General Information Manual, GA27-3345

IMS/ESA Version 6 Library

SC26-8725	ADB	Administration Guide: Database Manager
SC26-8730	AS	Administration Guide: System
SC26-8731	ATM	Administration Guide: Transaction Manager
SC26-8727	APDB	Application Programming: Database Manager
SC26-8728	APDG	Application Programming: Design Guide
SC26-8726	APCICS	Application Programming: EXEC DLI Commands for CICS and IMS
SC26-8729	APTM	Application Programming: Transaction Manager
SC26-8732	CG	Customization Guide
SC26-9517	CQS	Common Queue Server Reference
SC26-8733	DBRC	Database Recovery Control Guide and Reference
LY37-3731	DGR	Diagnosis Guide and Reference
LY37-3732	FAST	Failure Analysis Structure Tables (FAST) for Dump Analysis
GC26-8736	IIV	Installation Volume 1: Installation and Verification
GC26-8737	ISDT	Installation Volume 2: System Definition and Tailoring
SC26-8740	MIG	Master Index and Glossary
GC26-8739	MC	Messages and Codes
SC26-8743	OTMA	Open Transaction Manager Access Guide
SC26-8741	OG	Operations Guide
SC26-8742	OR	Operator's Reference
GC26-8744	RPG	Release Planning Guide
SC26-8767	SOP	Sample Operating Procedures
SC26-8769	URDB	Utilities Reference: Database Manager
SC26-8770	URS	Utilities Reference: System
SC26-8771	URTM	Utilities Reference: Transaction Manager

Supplementary Publications

GC26-8738	LPS	Licensed Program Specifications
SC26-8766	SOC	Summary of Operator Commmands

Bibliography

	Online Softcopy Publications	
LK3T-2326	CDROM	IMS/ESA Version 6 Softcopy Library
SK2T-0730	CDROM	IBM Online Library: Transaction Processing and Data
SK2T-0710	CDROM	MVS Collection
SK2T-6700	CDROM	OS/390 Collection

Index

Special Characters

Numerics

A

B

C

D

E

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Program Number: 5655-158

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-8730-04

Spine information:

IMS/ESA

IMS/ESA V6 Administration Guide: System

Version 6