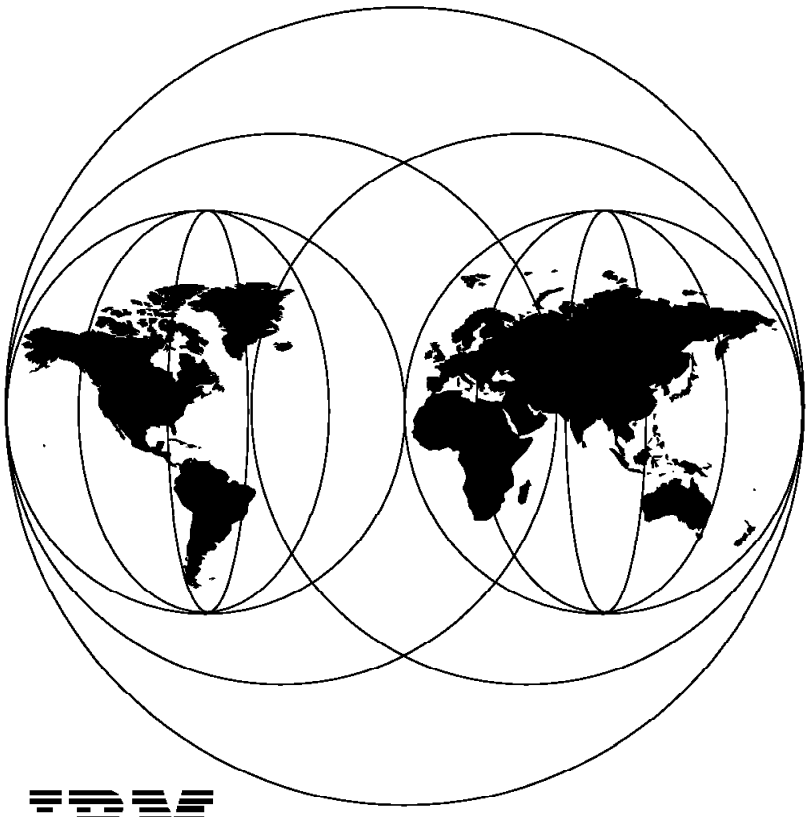
International Technical Support Organization

GG24-4302-00

IMS/ESA Version 5.1 Guide

February 1995



IBM

International Technical Support Organization
San Jose Center

International Technical Support Organization

GG24-4302-00

IMS/ESA Version 5.1 Guide

February 1995

Take Note!

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

First Edition (February 1995)

This edition applies to Version 5, Release 1 of IMS/ESA, Program Number 5695-176 (including the Database Manager, Transaction Manager, and Remote Site Recovery features), for use with the MVS/ESA Operating System.

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

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

IBM Corporation, International Technical Support Organization
Dept. 471 Building 070B
5600 Cottle Road
San Jose, California 95193-0001

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

Abstract

This document is unique in its detailed coverage of IMS/ESA Version 5.1. It provides information about the new features of this release, with special emphasis on their use and value in customer environments.

It also summarizes where IMS is positioned, especially for distributed processing and client/server solutions, and indicates how IMS, in conjunction with partner products, can meet present and future customer requirements.

This document addresses the needs of many different users of IMS. It contains information of interest to systems and application programmers, database administrators, and IMS operational staff. Chapters 1, 2, and 8 are also intended for technical management and planning personnel. A knowledge of IMS concepts and familiarity with earlier IMS versions are assumed.

(217 pages)

Contents

Figures

Tables

Special Notices

This publication will help executives and management understand the benefits of using IMS 5.1. For technical professionals, it describes the advantages of the new features of IMS 5.1 and will assist them in formulating an implementation plan for using the features and a migration plan for implementing this version.

The information in this publication is not intended as the specification of any programming interfaces that are provided by IMS/ESA 5.1. See the PUBLICATIONS section of the IBM Programming Announcement for IMS/ESA 5.1 for more information about what publications are considered to be product documentation.

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

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

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

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

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

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

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

CICS	COBOL/370
DB2	IBM
IMS Client Server/2	IMS CS/2

IMS/ESA
MVS/ESA
RACF
MQSeries
OS/2
RMF

The following terms are trademarks of other companies:

Windows is a trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

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

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to assist customers in their evaluation of IMS 5.1. It describes the benefits of the new features of IMS 5.1 as well as the processes and work effort required to migrate to IMS 5.1. It explains the value, benefits, and unique capabilities of remote site recovery and compares this new feature with alternative backup methods.

The content focuses on IMS 5.1, but the migration methodology and process descriptions can be used for any IMS version or release.

This document is particularly valuable for technical professionals (system programmers, administrators, planners, and operations staff) who are implementing an IMS system.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, “Introducing IMS/ESA Version 5 Release 1”

  This chapter provides an executive summary of IMS 5.1 and explains why the new release is interesting and valuable to many customers.

- Chapter 2, “IMS Version 5.1 Overview”

  An overview of the technical features of IMS 5.1 is presented.

- Chapter 3, “IMS 5.1 Processing Cost Enhancements”

  The new features of IMS 5.1 that are of particular interest for reducing the cost of processing transactions are described.

- Chapter 4, “IMS 5.1 Operating Cost Enhancements”

  The IMS 5.1 features affecting the operational costs of IMS are described.

- Chapter 5, “IMS 5.1 Availability and Remote Site Recovery”

  The new Remote Site Recovery feature of IMS is described, along with other changes in the base IMS product that improve availability and continuous operation.

- Chapter 6, “IMS 5.1 Distributed Processing and Open Access”

  Distributed transaction processing and open access to IMS from a large variety of platforms are of growing interest and importance. The excellent facilities provided in IMS 5.1 for other system access are described.

- Chapter 7, “IMS 5.1 Miscellaneous Enhancements”

  Various small (but important) enhancements to IMS are described.

- Chapter 8, “The IMS/ESA 5.1 System”

  The capabilities and uses of IMS are put into perspective. The wide range of capabilities provided by IMS often go overlooked, and this chapter gives a sampling of the many powerful uses of the product.

- Chapter 9, “Migration to IMS 5.1”

The migration of an IMS system is an important and potentially complex process. For IMS 5.1 this process is significantly simpler and easier than for past IMS releases. This chapter provides an overview of the migration process and describes the key technical considerations.

- Chapter 10, “IMS 5.1 and DBCTL”

  The use of DBCTL (and its importance for CICS users with the withdrawal of local DL/I support) is described.

- Chapter 11, “IMS 5.1 N-Way Data Sharing”

  An overview of using IMS in a parallel transaction server environment is provided as an introduction to the implementation detail in *Using IMS 5.1 with the Parallel Transaction Server*, GG24-4303.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document. Some of the publication listed below are not available at this time, but will be at a later date.

- *IMS/ESA V5 Application Programming: TM, SC26-8017*
- *IMS/ESA V5 Application Programming: Design Guide, SC26-8016*
- *IMS/ESA V5 Application Programming: DB, SC26-8015*
- *IMS/ESA V5 Administration Guide: DB, SC26-8012*
- *IMS/ESA V5 Administration Guide: System, SC26-8013*
- *IMS/ESA V5 Administration Guide: TM, SC26-8014*
- *IMS/ESA V5 Operator's Reference, SC26-8030*
- *IMS/ESA V5 Operations Guide, SC26-8029*
- *IMS/ESA V5 Failure Analysis Structure Tables, LY27-9621*
- *IMS/ESA V5 Utilities Reference: DB, SC26-8034*
- *IMS/ESA V5 Utilities Reference: TM, SC26-8022*
- *IMS/ESA V5 Utilities Reference: System, SC26-8035*
- *IMS/ESA V5 Release Planning Guide, GC26-8031*
- *IMS/ESA V5 Customization Guide, SC26-8020*
- *IMS/ESA V5 Messages and Codes, SC26-8028*
- *IMS/ESA V5 Installation Vol. 1: Installation and Verification, SC26-8023*
- *IMS/ESA V5 Installation Vol. 2: System Definition and Tuning, SC26-8023*

International Technical Support Organization Publications

- *Using IMS 5.1 with the Parallel Transaction Server*, GG24-4303
- *IMS 4.1 Migration Guide*, GG24-4150.

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

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

To get listings of ITSO technical bulletins (redbooks) online, VNET users may type:

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

How to Order ITSO Technical Bulletins (Redbooks)

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

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

Acknowledgments

This project was designed and managed by:

Attila Fogarasi
International Technical Support Organization, San Jose Center

The authors of this document are:

Hélène Lyon
IBM France

Alan Cooper
IBM UK

Jim Boyle
IBM Australia

Attila Fogarasi
IBM Santa Teresa Laboratory

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

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

Norma Andreadis
Fred Bucci
Dick Hannan
Mark Harbinski
Bob Huddleston
Barbara Klein
Sherry Li
Khi Mao
Rosmarie Martin
Dave Moore
Michael Morrison

Harry Radke
Frank Ricchio
Dave Thomas
Pat Schroeck
Terry Walker
Jack Wiedlin
Chung Wu
Mark Ziebarth
IBM Santa Teresa Laboratory

Bob Millar
IBM Hursley Laboratory

Bill Keene
IBM Dallas

Ed Berkel
IBM Poughkeepsie

Pete Sadler
IBM UK

A special thanks to our editor, Maggie Cutler, for her excellent work in editing this document to improve its clarity and style.

Chapter 1. Introducing IMS/ESA Version 5 Release 1

1993 was the year when IMS deservedly celebrated its twenty-fifth birthday. However, the main thrust of IMS is not looking back, but looking forward into the new world of the twenty-first century.

For many IMS customers, their view of their future systems has already undergone radical change. Almost without exception, customers are becoming far more cost sensitive than ever before. They want more computing power, more transactions per second, and more complex transactions, but with a reduced unit cost. Distributed and open computing is becoming the standard requirement. Many customers are requiring even greater levels of availability than in the past, and for some this means implementing remote site contingency facilities. Most customers, however, have many years of investment in IMS applications and skills, and a key requirement for them is that this investment should continue to be exploited in the future.

In summary, the key attributes of the “new world” are:

- Sensitivity to cost
- Very high availability
- Remote site contingency
- Distributed and open applications
- Enhancing or adding value to existing applications.

Clearly there is much in common between these attributes and the traditional values of IMS, namely:

- High performance
- Low cost per transaction
- Continuous operations and high availability
- Data and message integrity
- Device and terminal independence
- Application upward compatibility.

IMS 5.1 takes several new steps in helping customers achieve their objectives for the future:

- **Cost reduction**

  IMS 5.1 includes several enhancements that provide the potential for reducing both the execution and operating costs of the IMS system.

  The IMS 5.1 database manager can exploit an MVS Parallel Sysplex, running on multiple MVS systems in parallel, but all sharing the same databases. This allows some or all of the processors to be the newer CMOS-based machines, with their attendant cost benefits.

  IMS 5.1 Fast Path includes several major enhancements to its data entry database (DEDB). There is an option for selected DEDBs to be kept in memory. The High Speed Sequential Processing (HSSP) option has been redesigned to run even faster, and the HSSP Image Copy option has been replaced by a far simpler and much more usable version. The DEDB online

reorganization utility has been made to run much faster than before, but with greater efficiency.

IMS 5.1 interfaces to the new MVS Work Load Manager introduced in MVS 5.1, thus allowing transaction and batch performance goals to be specified in a meaningful, business-related manner. MVS monitors the performance against these goals, tunes itself as appropriate, and RMF then reports on how well the goals have been achieved.

There is a new and improved Automated Operator facility, which can be used in any IMS system including a DBCTL system.

Other items that help reduce operational costs include a more flexible time stamp recovery capability, more efficient processing for transactions that access DB2 databases, OSAM support for Dynamic Cache Management Enhanced (DCME), and yet more virtual storage relief in the CSA.

- **Higher availability**

  IMS 5.1 includes facilities for increasing the availability of IMS itself, databases, and even MVS.

  Problems with VTAM MSC links should no longer require IMS to be stopped and restarted. A new command parameter is provided to allow link problems to be corrected online.

  The enhanced time stamp recovery capability facilitates database recovery after application or operational errors and so minimizes the amount of time during which the data is offline.

  MVS must contain an SVC module for IMS. Previously, changing this SVC required MVS to be stopped and restarted. IMS 5.1 includes a utility to dynamically change the SVC without stopping MVS.

- **Remote site contingency**

  Remote Site Recovery (RSR) is a new feature of IMS 5.1 that extends IMS system and data recovery to a remote site. With RSR, IMS leads the way in high availability technology by providing automated transport, storage, and processing of log data and recovery information. An option of RSR is to maintain “shadow databases” at the remote site that are kept “almost current” by applying the updates from the log as they arrive. Alternatively, the remote IMS can simply collect the log data, and in the event of a takeover, the databases must first be recovered. In either case, the remote databases are updated only with complete sets of log records, and so consistency and integrity are assured.

  The techniques used by RSR are designed to minimize processor and I/O (and hence the cost) at the remote site. Similarly, because only log data is transmitted (and not all of the DASD updates), the communication costs are also minimized.

  With the use of database replication (shadowing), customers can reduce disaster recovery from days (or weeks) to minutes. Further, RSR can be used in a planned move of the IMS service to the remote site.

  A key feature of RSR, which distinguishes it from other techniques, is its ability to tolerate a link failure and catch up later using asynchronous log transport. This process is done very efficiently and without disrupting the real-time data transfer. RSR′s ability to minimize the impact on the running system even under overload conditions, while keeping the data as current as possible at the remote site, is an important and valuable feature that

provides higher levels of contingency protection at a lower cost than other backup techniques.

IMS RSR supports IMS DB/DC, DBCTL, and batch subsystems and works with Extended Recovery Facility (XRF) and data sharing configurations.

- **Open and distributed systems**

  IMS 5.1 provides a new high-performance interface between IMS and MVS applications that in turn provides standard communications facilities with IBM and non-IBM networks. This IMS facility, called Open Transaction Manager Access (OTMA), can communicate with any MVS subsystem or application program that implements the OTMA interface. Because standard MVS facilities such as the Cross-System Coupling Facility (XCF) are used, the implementation of the OTMA interface in other systems that want to communicate with IMS is relatively easy and straightforward.

  OTMA, used with new MVS OpenEdition application servers, enables very efficient access to existing transactions by users of TCP/IP, DCE/RPC, MQI, and other communications facilities. (These communications facilities already can be used to access IMS, but OTMA enhances the transaction options and the performance).

  IMS 5.1 provides POSIX compliance. Applications conforming to POSIX.1 standards can be easily ported to and from an MVS/ESA and IMS Transaction Manager (TM) environment. The POSIX.2 shell and utilities allow programmers with UNIX and C skills to develop programs for IMS through a familiar interface.

  The APPC facilities of IMS have been further enhanced by providing MSC support for LU6.2 devices. Customers who were unable to use APPC/IMS in IMS 4.1 because of their use of MSC can now implement native IMS APPC support. (Similarly, the DEDB Virtual Storage Option (VSO) removes the obstacle to APPC that Main Storage Databases (MSDBs) previously created.)

  MSC in general has been enhanced, and the routing exits have been rationalized and given greater flexibility. They support messages arriving both from traditional VTAM terminals and LU6.2 devices, as well as through OTMA.

- **Added value with protected investment**

  A major objective of IMS 5.1 is to enable existing applications to continue to provide value to customers who are implementing new terminals, communication models, networking facilities, and client/server solutions. OTMA is the obvious example. But APPC/IMS has also been enhanced to allow remote APPC applications to exploit the message mapping facilities that are inherent in the traditional 3270 IMS applications.

Fulfilling the Vision

IMS 5.1 provides the opportunity for customers to fulfill their vision for the future. It offers a level of flexibility and choice that is indeed impressive. It can be run in a single or a parallel processing environment. From a data perspective, the user has a rich choice of databases, including DB2, the various full-function DL/1 databases, and Fast Path DEDBs with the new data-in-memory option. With this full spectrum for performance, flexibility of use, and data availability, users can choose the database that best meets their requirements for processing cost, access paths, and dynamic change.

IMS system availability is second to none, and with this release, IMS enables system managed recovery at a remote location.

In IMS 5.1, IMS transactions can be accessed from a wide variety of both IBM and non-IBM platforms. All three open communications models are supported (APPC, DCE/RPC, and Message Queueing) together with the more traditional network accesses through VTAM. And of course, included in the IMS family of products are IMS Client Server/2 and IMS Client Server for Windows, which enable existing applications designed for use on 3270 terminals to be invoked from PS/2s, but with all of the added-value facilities that are available on intelligent workstations.

The world of computing is changing faster than ever. IMS will continue to change to keep abreast of the environment in which it runs. The IMS development team in IBM′s Santa Teresa Laboratories, in conjunction with other IBM laboratories in Poughkeepsie, Raleigh, and elsewhere, is working on making IMS better still, targeting the “new world” but without compromising traditional IMS values. IBM is determined that IMS will remain the unsurpassed database and transaction manager for the MVS platform.

Chapter 2. IMS Version 5.1 Overview

This chapter describes the new features of IMS 5.1. They are grouped as follows according to the potential benefit they can provide:

- Processing cost enhancements
- Operating costs enhancements
- Availability and remote site contingency
- Distributed processing and open access
- Miscellaneous.

2.1 Processing Cost Enhancements

The IMS 5.1 enhancements serve to reduce the unit cost of processing a transaction. Customers primarily interested in reducing costs should investigate migrating to IMS 5.1, focusing on the relevance to their environment of the features discussed below.

2.1.1 Sysplex Support and N-way Data Sharing

An MVS Sysplex is a set of loosely coupled MVS systems whose objective is to present a single system image to the end user. Here, the end user could be a terminal user, a client program, or a submitter of a batch job.

With a sysplex approach customers can:

- Have the required CPU power through multiple CMOS microprocessors, with their inherent cost benefits
- Avoid capacity limits of a single processor
- Grow incrementally (and hence reduce costs) to meet increased capacity requirements
- Experience very high availability
  - Processors can be added or removed dynamically
  - Software maintenance can be applied online (on one processor at a time)
  - The loss of one component only proportionately reduces the system capacity. The workload can be redistributed across the remaining components.
- Use ES/9000 processors in a sysplex, so allowing their full value to be realized.

An MVS Sysplex, which includes the use of the new microprocessor hardware and is well-suited to provide transaction processing, is referred to as a *Parallel Transaction Server (PTS)*.

A key component of a parallel sysplex is the *Coupling Facility (CF)*, which provides storage and function to allow resources to be efficiently shared by all the MVS systems.

IMS 5.1 is the first IMS release designed to exploit the parallelism of a PTS system. In this release, IMS databases can be shared by all the IMS systems (transaction processing or batch), as shown diagrammatically in Figure 1 on page 6.

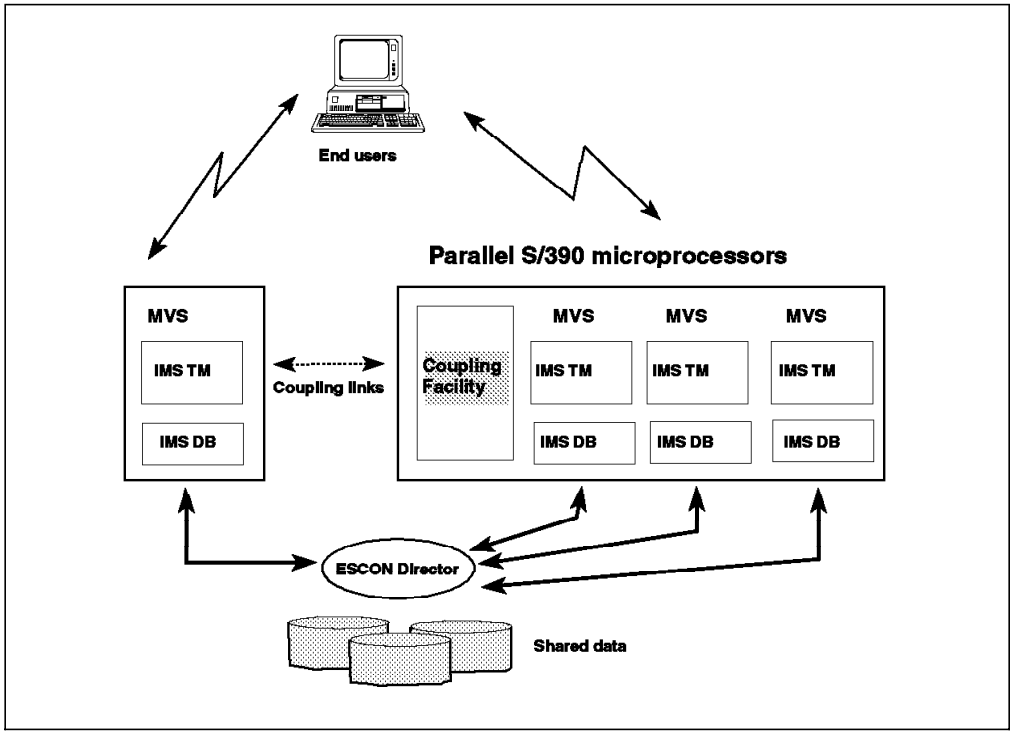


Figure 1. Parallel Transaction Server

This IMS facility is called *n-way data sharing*, and is a natural extension of block level data sharing (BLDS). BLDS was limited to a maximum of two MVS systems, whereas n-way data sharing can accommodate a maximum of 32 sharing subsystems.

The fundamental component of a data sharing facility is the lock manager, and the IMS 5.1 n-way data sharing facility includes a new version of the *IMS Resource Lock Manager (IRLM)*, designed to work in conjunction with the CF. Each IMS subsystem has its own IRLM. Unlike the previous two-way BLDS, however, the new IRLMs do not communicate directly with each other through VTAM; instead they maintain control information in the CF. This arrangement simplifies IRLM implementation and enables consistently high locking performance regardless of the number of sharing subsystems.

Each IMS database manager maintains its own local buffer pool. When a buffer in one system is updated, that same data may sometimes be present in another IMS system′s buffer pool (because that was where it had previously been processed). In that case, the updating IMS system notifies all systems that also have this data in their pools that their copies are no longer valid. This process is called *buffer invalidation*, and, although initiated by the updating IMS, it is the CF that (a) recognizes which systems have the data and (b) directly flags the relevant buffers as invalid. (Buffer invalidation is not necessary for Fast Path DEDBs.)

The other major component of the IMS n-way data sharing facility is Database Recovery Control (DBRC). Customers should already be using DBRC at the Share Control Level to reap the benefits of system managed database integrity, even in a nonsharing environment. All shared databases must be registered with DBRC. DBRC then authorizes each shared access, tracks the multiple logs being produced on the various IMS systems, and should be used to generate the appropriate recovery jobs when needed.

2.1.2 Fast Path Enhancements

IMS 5.1 contains several enhancements to Fast Path DEDBs. Mostly these are performance enhancements, but a major usability enhancement is the option to replace Fast Path MSDBs with an in-memory DEDB.

The VSO allows any areas of a DEDB, subject to available real storage, to be loaded into an MVS data space. There is an implementation option to read the area into memory either at open time or as each control interval (CI) is first referenced. Once a CI has been read into memory, all application or utility reads will be from the data space. When a CI is updated, it is written to the data space and the lock is released immediately. Periodically, IMS writes out updated CIs from the data space to the area data sets on DASD.

VSO will be attractive to customers with very heavily utilized areas. It also is intended to provide a replacement for MSDBs, which have for many years offered exceptionally high performance, including the ability for multiple applications to update the same data at the same time, with integrity (using the DL/1 field (FLD) call). However, the use of MSDBs was subject to certain restrictions, such as single segment hierarchy, fixed length segments only, no DBRC support, and unique backup/recovery procedures and utilities. VSO DEDBs can have the full hierarchy and use standard DBRC and backup/recovery facilities. To complement the VSO, DEDBs in general have been enhanced in other ways. Segments can be defined as fixed or variable length, and the DL/1 FLD call has been implemented for DEDB. (Note that these enhancements apply to all DEDBs, not just VSO DEDBs).

Other DEDB enhancements in IMS 5.1 include the option for selected areas to be preopened at IMS startup time (or /START command). An application program can use the Q command code to guarantee that no other program will update the CI until the calling program has reached its syncpoint. The DL/1 DEQ call can now be used with DEDB to invoke buffer stealing.

The DEDB Direct Reorganization Utility (DBFUMDR0) has been rewritten. This utility has long been unique in providing a truly online reorganization facility, but it was designed for function rather than for performance. The new utility, the DEDB High Speed Reorganization Utility, no longer uses the Reorg unit of work (UOW) but instead does all data reorganizing in memory. As a consequence, log data is significantly reduced. Memory is also used to enable full overlapping of I/O with processing. As its new name suggests, this utility in general runs much faster than in version 4.1, perhaps as much as 10 times quicker.

The HSSP option was introduced in IMS 3.1 to enable BMPs that do physical sequential processing to exploit the sequential caching facility of the 3990 DASD Control Units. Customer use of HSSP has resulted in mixed feelings about its benefits. In IMS 5.1, HSSP has been rewritten to have no hardware dependencies. Now it uses chained I/Os under all circumstances, does look-ahead buffering and asynchronous writes, and uses larger numbers of private buffers to make this possible. The result should be consistently high performance for all BMPs doing true sequential processing.

The original HSSP also included an Image Copy option, which enabled an image copy to be taken at the same time that an application processed sequentially through some or all of an area. However, many customers found this an unattractive option because the image copies had to be written to DASD, and in general dual copies were required (HSSP updates were not logged, but were

captured on the HSSP Image Copy). Further, care had to be taken to ensure that HSSP Image Copy data sets were not reused before being archived to some other backup copy. If an HSSP Image Copy application program failed before completion, there were nonstandard recovery implications (the partial image copy had to be treated as a different type of log), and, although DBRC could manage the recovery process, customers generally found the operational aspects to be error-prone and a rather complex area to handle.

IMS 5.1 contains a total replacement for the HSSP Image Copy facility. The new HSSP Asynchronous Image Copy (ASIC) facility allows a standard Concurrent Image Copy to be taken as the HSSP BMP is processing the area. The copy can be to tape or DASD. All area updates are logged, so there are no longer problems associated with applications abending—the incomplete image copy is simply discarded. In other words, IMS 5.1 removes the operational complexity associated with the HSSP Image Copy facility.

The Fast Path Log Analysis Utility (DFBULTA0) is commonly used to report on transaction and BMP performance, and in IMS 5.1 it has been significantly enhanced. Transit time components are now reported in milliseconds, and up to three optional report lines provide detailed statistics on DEDB calls, VSO, and buffering.

2.1.3 MVS Work Load Manager Support

IMS 5.1 has been designed to work with MVS/ESA SP Version 5.1 (MVS 5.1), which contains a new component called the Work Load Manager. To achieve user-specified performance goals, the Work Load Manager manages in an MVS Sysplex workload distribution, the balancing of workloads, and allocation of resources to competing workloads. The user uses a panel-driven application, supplied with MVS, to define the different workloads that will run at different times of the day or on different days of the week. The user categorizes batch work and transactions into service classes according to their importance to the business. The user defines how the Work Load Manager can tell to which service class any particular job or transaction belongs to (for example, according to transaction code or class, LTERM or USERID). The user also specifies the performance goal for each service class (for example, 95% of transactions complete in less than 1 second).

An online IMS 5.1 system keeps MVS 5.1 notified of which transactions it is scheduling and what the transactions are doing (processing, waiting for a lock or a database I/O, or terminating). The Work Load Manager samples this data and adjusts the machine resources under its control to try to meet the specified performance goals. If the system is overcommitted, the Work Load Manager ensures that the work with the highest business priority is processed first.

RMF has also been enhanced to report on performance within each service class and shows the achievement in terms of the goals.

The interface used to define the performance criteria is very much simpler than with previous releases of MVS. Consequently, it provides better control and reduced errors, resulting in a higher achievement of business objectives.

IMS 5.1 used with the MVS 5.1 Work Load Manager provides the opportunity for customers to review and simplify their IMS class scheduling structure, with the potential for improved performance at a reduced cost.

2.1.4 Miscellaneous

Pseudo-Wait-For-Input (P-WFI) was introduced in IMS 3.1 to reduce the number of program loads by exploiting all available dependent regions and not terminating a program until it was clear that its region was the best place to schedule another program. However, applications that accessed DB2 were unable to exploit this facility. When an IMS/DB2 program had no more messages to process, it was always terminated.

In IMS 5.1, P-WFI applies to IMS/DB2 applications, which can gain the benefits not just of reduced program load but also of reduced DB2 thread creation.

To reduce the CPU cost of accessing DB2, thread creation (at program schedule time) and SIGNON to DB2 (for every message) have been rewritten. They no longer use an SVC call, and they use the new MVS storage management facilities.

The IMS Log Analysis Utility, DFSERA10, has been enhanced with improvements to the ERA70 exit. If a problem requires an understanding of which database updates were done and by which application, for example, the new function in ERA70 will expedite the process of problem resolution.

Virtual storage constraint, especially in CSA, is still a concern for some customers. IMS 5.1 offers about 100KB relief in CSA.

OSAM databases, used on a cached DASD Controller, can now gain the benefits available with DCME and in particular record level caching. Online applications doing predominantly random processing against OSAM databases have the potential for significantly improved performance and more efficient use of the 3990 DASD Control Unit.

2.2 Operating Cost Enhancements

The operational aspects of IMS are enhanced for online and DBCTL environments. The changes are small but far reaching. Existing automation scripts may benefit from revision in view of the changes in the messages.

2.2.1 New Automated Operator Facilities

IMS 5.1 provides an automated operations tool for all IMS users, including DBCTL customers. Before IMS 5.1, the only facilities available for DBCTL were those provided independently of IMS by products such as NetView.

For IMS DB/DC users, the new Automated Operator facility represents a third tool to help automate operations:

- Time Controlled Operations (TCO), available for DB/DC and DCCTL environments
- Automated Operator Interface (AOI) - Type 1, available for DB/DC and DCCTL environments
- Automated Operator Interface (AOI) - Type 2 (new in IMS 5.1), available for DB/DC, DBCTL, and DCCTL environments

2.2.2 New Master Terminal Opterator Facilities

A number of operator enhancements have been added to IMS 5.1, especially to assist in the management of databases.

Databases can now be allocated to groups, and the standard database commands (/DBR, /START, /STOP, /DBD) can be issued against a group. Fast Path DEDB areas can also be members of groups.

A /START DB command can now be expedited by requesting that data set allocation be deferred until first DL/1 call time.

When commands are entered against a group of (or all) databases, a single response is returned to the terminal, instead of one message per database (any problems are still reported individually).

Using the ALL parameter on database commands has until now been generally acceptable, with one possible exception, the /DISPLAY command. Consequently, with IMS 5.1 the facility for disabling ALL on selected commands (not just database commands) has been enhanced to allow the specific exclusion of ALL on /DISPLAY commands.

2.2.3 Enhanced DBCTL Operational Interface

A DBCTL system is operated from an MVS console. IMS 5.1 has enhanced this interface by substantially reducing the number of messages sent to the console, and by allowing the DBCTL system ID to be used instead of an arbitrary command recognition character.

2.3 Availability and Remote Site Contingency

The availability enhancements are in the base IMS 5.1 component and result in significant availability improvements for specific resources (such as MSC). These are very important changes for the affected environments. MSC users are particularly urged to evaluate the Message Control/Error Exit (DFSCMUX0) for their systems.

The remote site backup capability is part of the separately priced Remote Site Recovery feature of IMS 5.1. This feature can be used with (or without) XRF to provide a high level of protection against a disastrous outage. RSR is not a replacement for XRF and protects against longer term failures than XRF.

2.3.1 Remote Site Recovery Feature

The RSR feature of IMS 5.1 provides remote recovery for IMS DB full function databases, IMS Fast Path DEDBs, and the IMS TM message queues. There are no distance limitations with IMS RSR.

The success of many companies depends on the availability of their computer services. Indeed, for some companies, their very survival depends on their computer services being almost permanently available. The computer service must remain, even when a computer center itself suffers an extended outage. A typical RSR configuration is depicted in Figure 2 on page 11.

RSR is the ability to quickly reinstate the critical workload at a remote computing center after a loss of service at the active site. The remote site must be far enough away not to be affected by the outage.

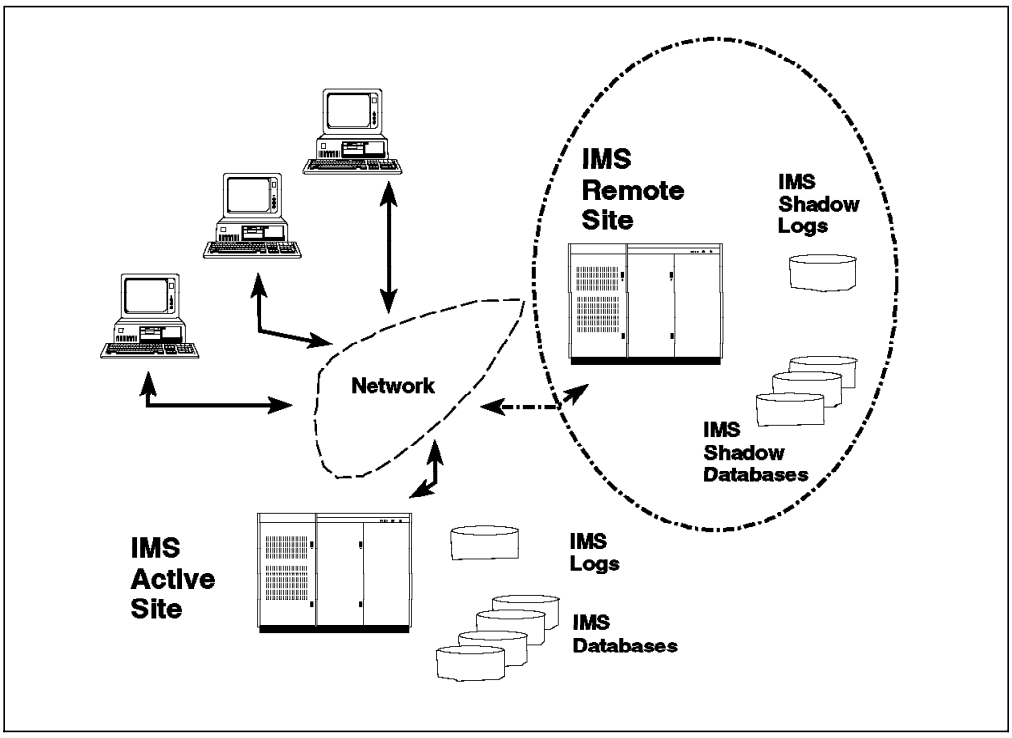


Figure 2. Remote Site Recovery

Nothing is shared between the two systems; only a communications line is needed to transmit log data from one site to the other. The remote site, also referred to as the tracking site, remains ready to take over the work of the active site at any time. This transfer of service can be planned or unplanned.

A takeover involves shutting down the tracking subsystem, performing forward recovery for those databases that require it, and then starting the remote IMS subsystem as the new active subsystem to run the critical production work.

The key function used by RSR to achieve this objective is the sending of log data to the tracking site as each log buffer is filled. The database changes arrive at the tracking site almost as soon as they occur at the active site. Some recovery-related information (DBRC data) is added to the log to ensure that the data transmitted to the tracking site is sufficient for full recovery with integrity. The databases that are to be tracked must be registered in DBRC.

Normal IMS processing is not impacted by the RSR implementation, and the recovery process does not need application program changes. RSR uses standard, though enhanced, recovery procedures. A particularly important point regarding data integrity is that RSR always provides database consistency at the remote site. Only complete sets of updates (units of recovery) are applied to the remote databases.

Those customers who are using XRF can also use RSR. The two techniques complement each other and remain consistent with a continuous availability strategy.

It is important to recognize that for customers wanting remote site contingency, RSR is just one component of the total solution. It covers the application DL/1 data, but not program libraries, sequential files, and other types of database.

2.3.2 Enhanced Time Stamp Recovery

IMS 5.1 introduces a new identifier that uniquely relates to a time period during which a database is continuously open for update. This identifier is written in the log records for the database, and it is recorded by DBRC. Consequently, DBRC can now enable time stamp recovery to any point on a log where the database was not open for any users. Timestamp recovery is no longer dependent on the use of FEOV on /DBR commands.

2.3.3 Enhanced Message Control/Error Exit

The MCEE was introduced in IMS 4.1 for MSC and has been extended in IMS 5.1 to become the focal point for managing both MSC and APPC communication failures. It is used to minimize the impact of failures and to address any message integrity exposures. The APPC/IMS Destination Exit, DFSLULU0, has been replaced by the MCEE.

2.3.4 MSC Forced Close of Link

Before IMS 5.1, certain problems with VTAM MSC links could be resolved only by restarting IMS. In IMS 5.1, the /STOP command has been enhanced to enable the IMS and VTAM perspectives on the session status to be reconciled online.

2.3.5 Dynamic Update of IMS Type 2 SVC

IMS 5.1 includes a new utility that allows an IMS SVC to be added to or replaced in an MVS system without re-IPLing the system. This is one further facility for preventing a scheduled outage of the MVS system, which can be important to satisfy user demand for continuous operations.

2.4 Distributed Processing and Open Access

The ability to access any system, any data, any time is of growing and often-times crucial importance. In IMS 5.1 both brand new facilities and enhancements to existing facilities provide a wide variety of access. IMS becomes one of the most open subsystems in the world with support for all transaction models and an architecture that facilitates incorporation of new MVS capabilities as they become available.

2.4.1 Open Transaction Manager Access

OTMA provides a client/server protocol between IMS applications and other MVS programs. The MVS programs are typically intermediate servers to end users on workstations using communications facilities such as TCP/IP and communication models such as APPC, Remote Procedure Call, or Messaging and Queueing. The main objective of OTMA is to simplify the communication between the MVS programs (IMS OTMA clients) and the IMS server for transaction processing. OTMA is a new interprocess communication facility within an MVS environment.

OTMA is implemented in an MVS Sysplex environment and is based on the MVS XCF, a way of exchanging data between two processes at high speed, introduced in MVS/ESA Version 4. IMS communicates with these MVS applications over an XCF group. The members of an XCF group are the MVS application programs (the OTMA clients) and IMS itself.

OTMA runs only in IMS TM or DCCTL environments and not in a DBCTL environment. Because existing IMS applications can run without modification and be called from any OTMA client, there is more flexibility in the choice of routes to access IMS applications.

Figure 3 on page 14 illustrates an IMS system with OTMA.

Several other IBM products will exploit OTMA at the earliest possible opportunity:

- MQSeries for MVS/ESA

  With OTMA, existing IMS applications will be available in a network using MQI as the communications interface. The MVS MQ queuing will be bypassed, and only IMS queuing will occur.

- IMS TCP/IP

  This MVS product allows workstations to enter transactions by means of TCP/IP into IMS. There are plans for a subsequent release to enable TCP/IP sockets to interface with existing IMS applications using OTMA.

- DCE Application Server/IMS (AS/IMS)

  RPC calls can access existing IMS applications with AS/IMS, using ISC as the communication interface.

  A new version of AS/IMS will use OTMA instead of ISC to take advantage of the simpler interface and improved performance.

The OTMA interface is externally documented, enabling user or vendor applications to be created according to the market demand. But note that the design and coding of an OTMA client require skills that are definitely not within the scope of a typical COBOL application programmer.

OTMA offers the following advantages:

- High performance access to IMS using the MVS XCF application programming interface (API)
- Existing applications run without change
- No system definition dependencies
- Freedom from networking architecture
- Flow-control and transaction attributes can be dynamically bound to the transaction
- Full-duplex processing.

OTMA uses a Tpipe instead of an LTERM. A Tpipe is a named IMS process management resource that the OTMA client must specify when submitting a transaction to IMS. It is a logical structure that represents an anchor point for client transaction input and output . The Tpipe name is unique within a client structure in IMS.

This OTMA protocol treats transactions as data objects that have attributes independent of application, session, or transport layer considerations and encapsulates all transaction data within the OTMA protocol messages. It also tries to keep a minimum amount of information in the IMS control blocks by including the information in the message itself.

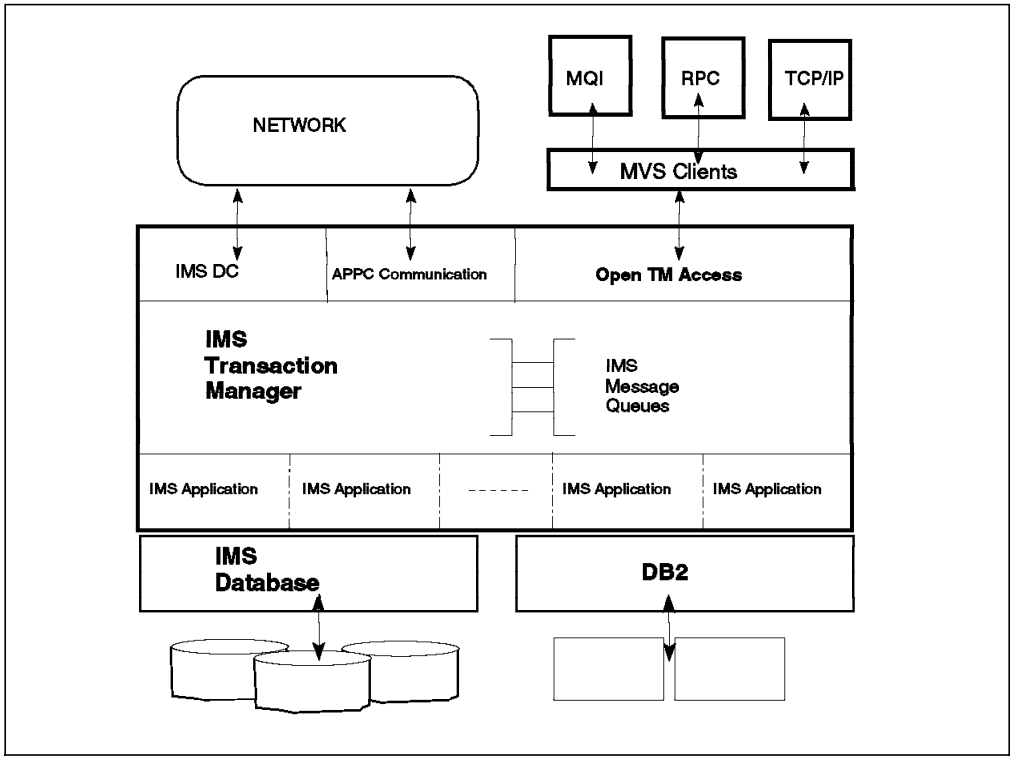


Figure 3. Open IMS

OTMA supports non-response-mode and response-mode transactions, including IMS conversational and Fast Path transactions.

As for APPC/IMS, OTMA does not support MFS.

2.4.2 POSIX Compliance

IBM has chosen to integrate the open interfaces for systems services into the MVS operating system. IBM OpenEdition Services for MVS/ESA V4R3 now provides a common set of system support services based on a series of Portable Operating Systems Interface (POSIX) standards.

Because IMS exploits the facilities of MVS, it will get these POSIX services without additional changes to IMS. In other words, POSIX standards are now supported in IMS applications.

Figure 4 on page 15 shows the architecture of MVS OpenEdition.

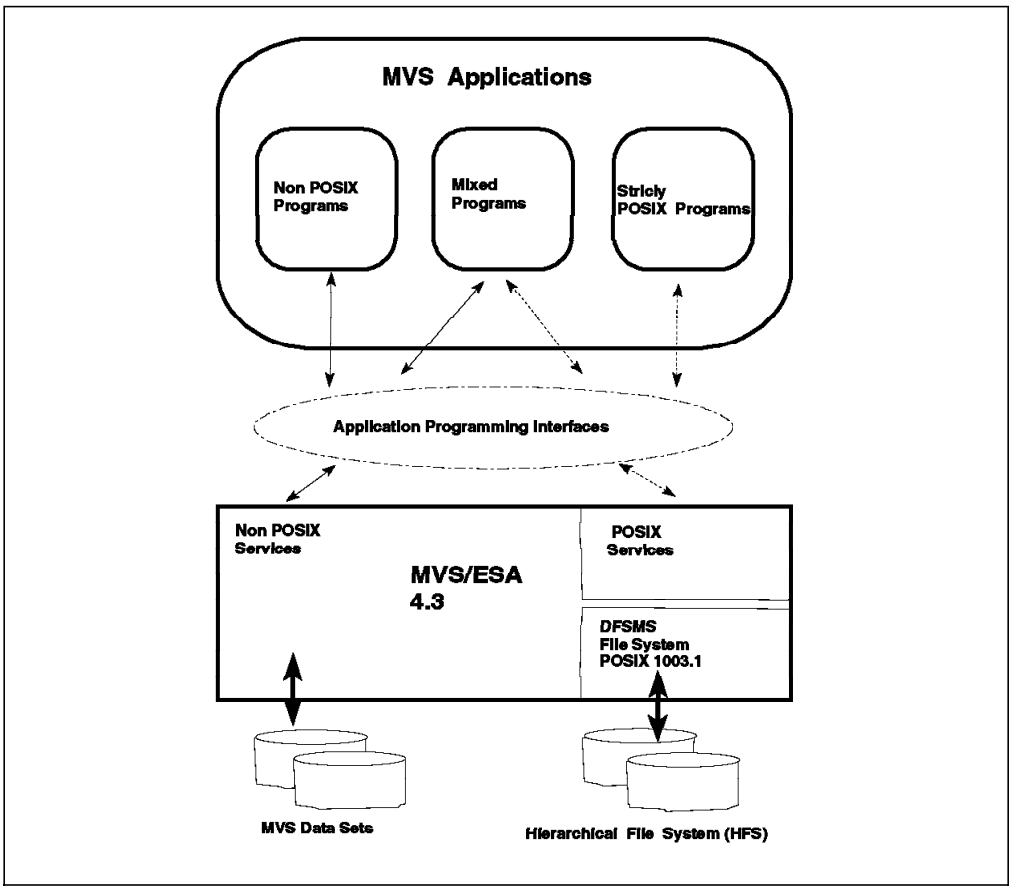


Figure 4. MVS OpenEdition

2.4.3 Enhanced APPC/IMS

The APPC/IMS support introduced in IMS 4.1 has been enhanced in IMS 5.1 as follows:

- Transactions entered through APPC can make full use of the IMS MSC facilities. The transaction can be processed in a remote IMS system, or if the transaction is local it can use an alternate PCB to send a message through MSC to a remote system. A reply to the IOPCB in a remote system will be routed back to the originating LU6.2 application. In other words, all of the restrictions in IMS 4.1 have been removed.
- The functions of the LU6.2 Destination Exit (DFSLULU0) have been implemented in the MCEE (DFSCMUX0). The LU6.2 Destination Exit (DFSLULU0) no longer exists in IMS 5.1.
- The Terminal Routing Exit (DFSCMTR0) is functionally replaced by the new Input Message Routing Exit (DFSNPRT0), which is called for all input messages, including those originating from an LU6.2 terminal.
- Qualified Network LU names are now supported in inbound LU6.2 conversations.
- There is a new message mapping facility.

The LU6.1 adapter is no longer supported.

2.4.4 MSC Enhancements

In IMS 5.1, the message routing exits have been enhanced and rationalized. Messages arriving from any terminal, LU6.2 device, or OTMA can be rerouted to any local or remote destination. Likewise, any messages sent to an alternate PCB by any application, local or remote, can be rerouted to any destination, local or remote.

In a Parallel Sysplex environment, certain transactions can be routed to a specific target system by using the new MSC exits without complex system definitions.

2.5 Miscellaneous Enhancements

Other facilities introduced in IMS 5.1 include additional access security checks for online systems using VSAM databases and DEDBs.

Also, for most customers, the new MVS four-digit device addresses will be implemented transparently by IMS 5.1.

2.6 Discontinued Support

For a long time now, IBM has indicated that CICS Local DL/1 would not be supported after IMS 4.1. All CICS DL/1 users must convert to using DBCTL before migrating to IMS 5.1.

The LU6.1 adapter for LU6.2 applications will not be supported in IMS 5.1. The adapter is shipped with IMS 5.1 and can be used on an unsupported basis at the customer′s own risk. This may ease the migration burden for some customers; in general a conversion to APPC/IMS is recommended before migrating to IMS 5.1.

Chapter 3. IMS 5.1 Processing Cost Enhancements

This chapter describes the IMS 5.1 enhancements that relate to IMS performance, especially those that provide the potential to reduce the processing costs for an IMS system.

3.1 Sysplex Support and N-way Data Sharing

The reasons to use data sharing are many. They include:

- Throughput requirements exceed the capacity of any single system
- Desire to improve data availability
- Need for differing workloads at different times of day without disrupting any other workload
- Desire to change database buffer sizes without shutting down IMS.

IMS data sharing adds considerable capability and flexibility to an environment. At the same time, the operational environment is significantly more complex due to the additional components, and the data sharing implementation plan must address these changes.

The MVS Work Load Manager support is also significant in the successful operation of a sysplex. IMS 5.1 interfaces to the new Work Load Manager component allowing the tracking and control of every single transaction in the IMS system.

N-way data sharing provides data concurrency control, allowing full integrity of update and data access, at a reasonable cost. This objective presents some daunting challenges, which are met using a variety of algorithms that reduce the frequency and cost of lock contention.

Work Load Manager support provides for prioritization of workload and allocation of system resources to maximize the system performance for the customer.

The changes to n-way data sharing in IMS 5.1 are:

- A new IRLM, which can:
  - Connect to 31 other IRLMs
  - Use a new hardware function, the Coupling Facility, to provide speedy communication between IRLMs
- A new locking scheme within IMS
- Propagation of buffer invalidation using the Coupling Facility when a buffer content has been updated by a different system.

3.1.1 Implementation

The implementation is transparent to the installation and is largely self managing. This is a key attribute of the system. IRLMs and IMSs can be dynamically started or stopped, using the data sharing group, without affecting the other members of that group.

3.1.2 Value

The value of running IMS 5.1 in a sysplex is twofold:

- The system can grow to exceed the limits of a single system without sacrificing data currency and without data replication (and the attendant problems of that technique)
- Smaller building blocks for the aggregate system can be used; this is particularly useful for the 9672 PTS machines.

IMS n-way data sharing in a 9672 PTS significantly reduces the cost per transaction while providing extremely high service levels in terms of availability, integrity, reliability, and performance. For value conscious customers IMS is the transaction manager of choice!

3.2 Fast Path Enhancements

3.2.1 Virtual Storage Option

VSO is a high performance option for DEDBs and the preferred alternative to MSDBs for holding data in memory.

VSO is specified in DBRC for a DEDB area. It causes that area to be loaded from DASD into virtual storage using an MVS data space. Subsequent reads are done from the data space without I/O. Updates are written at application syncpoint time to the data space. Periodically, IMS copies updated CIs from the data space to the area data sets on DASD.

VSO Use of Data Spaces

VSO areas are mapped to MVS data spaces. When IMS starts up, if Fast Path is generated in the system (FPCTRL macro specified in IMS system definition), two 2GByte data spaces are acquired. One data space has the disabled reference (DREF) option specified, which means its pages can reside in main memory or expanded storage but will never be paged out to DASD. The other data space does not have the DREF option and so may experience paging to DASD if there is a shortage of real storage.

No actual storage is allocated when the data spaces are acquired, other than to hold the MVS control blocks. Only when the data spaces are used are virtual and real storage needed, and then only the amount that is actually required is allocated.

CIs are mapped linearly to a data space. In other words, the location of a CI in a data space is calculated by adding the CI RBA to the starting address of that area in the data space. Data spaces are initialized to binary zeros. To detect whether a CI is in the data space or not, Fast Path reads the appropriate data from the data space and checks the control information at the end of the CI. For a good CI, this control data is the RBA of the CI, the VSAM Record Descriptor

Field, and the VSAM CI Descriptor Field. Only if all three fields are what Fast Path would expect, does the copy in the data space get used. Otherwise (for example, all fields contain binary zeros), the CI is read from DASD and immediately copied into the data space.

Preload Options

Two types of VSO areas can be defined, preloaded and non-preloaded. These definitions are made in DBRC using CHANGE.DBDS (or INIT.DBDS as appropriate). Any area defined with the PRELOAD option will be read into the DREF data space following the IMS initialization system checkpoint. Only the Direct Part (that is, everything up to the end of IOVF) will be loaded. Application programs can be scheduled before the load completes, and a request for a CI that has not yet been loaded will result in a synchronous read I/O request. Data will be staged into a buffer in the Fast Path Common Pool, and the CI will then be immediately copied into the data space.

The (pre)load is done one UOW at a time; a benefit of DEDB using the Media Manager is that the load I/Os are done directly into the data space. A lock is taken on the UOW before it is loaded to avoid conflict with early programs that might want to update CIs in that UOW. It is also possible that one or more of the CIs in the UOW has already been brought into the data space and updated by an early program. So Fast Path, having read in the UOW, then checks each CI individually, to see whether it is in the data space. If it is, that CI is skipped. Otherwise, the loaded CI is copied to the data space.

If an area has the VSO NOPREL (non-preload) option, each Direct Part CI is allocated a position in the Non-DREF data space, but no automatic load will take place. The first time a CI is requested after area open, IMS will try to read it from the data space, but will find binary zeros. So IMS will read the CI from the area data set on DASD into a buffer in the Fast Path Common Pool and will then copy it into the data space. The next time the CI is requested, it will be found immediately in the data space. CIs that are never referenced will not get loaded into the data space. SDEP CIs are never loaded into the data space.

CIs from PRELOAD areas will always remain memory resident, in either main or expanded storage (because of the DREF option). CIs from NOPREL areas can be paged out to page data sets if there is storage constraint. MVS paging should be avoided, as the performance would almost certainly be worse than doing the I/Os to the actual area data set. The reason for providing the NOPREL/Non-DREF option is for cases where typically only a small part of an area is accessed. In this case, data space storage will only be needed for those CIs actually referenced.

Multiple Data Spaces

IMS uses a “best fit” technique for allocating areas to data spaces. If necessary, IMS will allocate additional 2GB data spaces, of either type. However, a VSO area cannot span more than one data space. Further, a few hundred KB of the data space are used for control purposes (for example, as a staging area for writing out updates as described below). Consequently, the maximum allowable size of a VSO area is a little short of 2GB.

Updating VSO Areas: When an application program updates a VSO CI, the update is held in a Fast Path buffer until syncpoint time, just as for non-VSO data. For non-VSO updates, physical logging of the Commit Log Record (5937) will drive the Output Threads (OTHREADs) that physically perform the database

write I/Os. This physical logging is an asynchronous event after syncpoint. In the case of VSO, the updates are made to the VSO data in the data space as part of syncpoint phase 2 processing. In other words, the database can be updated before it is physically logged. Log Write Ahead is not required with VSO because, in the event of a system failure, the updated data in the data spaces will be lost. Only when updates have been physically logged do they become recoverable. IMS ensures that updates made to the data space are not copied out to DASD until after they have been physically logged. In this respect, VSO does use Log Write Ahead, and data integrity is assured.

IMS needs to keep track of which VSO CIs have been updated. A bit map in the IMS Control Region is used for this purpose.

Once the updates have been copied into the VSO data space at syncpoint time, Fast Path releases the exclusive lock on the CI. This is a significant benefit compared with the non-VSO case, where the lock is held until the CI has been written to DASD, which follows the asynchronous physical logging.

Write I/O to the Area Data Sets

IMS periodically writes updated CIs from the data spaces to the area data sets. IMS uses the bit map to determine which CIs need to be written out and resets the bit when an I/O completes. All updated CIs from an area are written out. The frequency of write-back is determined by the size of the Direct Part of the area. For areas with 10,000 or less Direct Part CIs, writing to DASD takes place every four minutes. For larger areas, the write-back frequency is once per minute. In addition, the write-back of all updated CIs is initiated at each IMS system checkpoint. This is an asynchronous process and has no impact on checkpoint duration. Outstanding updates are, of course, also written out at Area Close.

The area data set writes are done by standard OTHREADs, which use chained I/O from a 200KB staging area that is page fixed in each data space. The I/Os are directly from the staging area in the data space to DASD. Before each chained I/O is started, a Wait Write to the Log is issued for the most recently created Fast Path Commit Log Record (5937). This ensures compliance with the Log Write Ahead requirement.

The set of updated CIs that is copied to the staging area (for example, 50 CIs of 4KB) can span two or more areas of one or more DEDBs, in which case multiple chained I/Os will be initiated in parallel. Also the two or more data spaces will be processed in parallel.

VSO and MADS

A VSO area can be implemented with MADS. There will only be one copy in the data space, but the write-outs will take place to all copies of the area on DASD. The usual rules for write I/O errors apply. If the second CI or more than 10 other CIs experience write I/O errors, the ADS is stopped. If the one and only ADS is stopped, the area becomes unavailable and is flagged as “recovery needed,” and the data space copy is released. Preloading an area does not protect it from DASD failure.

VSO and Utilities

The DEDB utilities, and their exploitation of VSO, are described in Table 1.

Table 1. DEDB Utilities and Their Exploitation of VSO

Utility	***VSO Exploitation***
Area Initialization	Not applicable. It always initializes an area on DASD and is run offline.
SDEP Scan	None. SDEPs are not loaded into a VSO data space.
SDEP Delete	None. SDEPs are not loaded into a VSO data space.
Online Reorg	All reads and writes are satisfied through VSO data space.
Area Create	Reads are from VSO copy. Writes are done only to new area data set on DASD.
MADS Compare	Not usable with VSO area. Before running Compare, it is necessary to unload the VSO data to DASD using the /VUNLOAD command.

Concurrent Image Copy works perfectly well with VSO areas but only reads the DASD copy of the data. DBRC will include all necessary logs in a GENJCL.RECOVER. Because of the write-backs initiated at each checkpoint, the standard rules for log selection apply equally with or without VSO.

The Recovery Utility will recover a DASD copy of the area. The subsequent /STA AREA will enable the data to be loaded into the data space.

Data Sharing with VSO

In IMS 5.1, VSO areas cannot participate in BLDS. In particular, this means that the n-way data sharing support, introduced in IMS 5.1, excludes VSO DEDBs.

VSO areas can be defined to allow Area Level Sharing (that is, registered in DBRC with SHARELVL of 1), but this should be used with caution. The first subsystem requesting DBRC authorization against a VSO area will exploit the VSO facilities. The subsequent subsystems will have to do all reads and writes directly to the data set.

If the first subsystem is an updating subsystem, by definition all other subsystems will be read-only. Because the updating system is using VSO, the read-only programs will see data on DASD that can be a few minutes out of date. During the periods of write-back from the data space to DASD, the read-only programs run the risk of reading partially updated database records or getting a GG status code.

If the first subsystem is a read subsystem (read, not read-only), it will exploit VSO, and all subsequent subsystems will read directly from DASD. Clearly, there are no issues in this case.

If the first subsystem is a read-only subsystem, it will acquire the VSO facilities. One of the subsequent subsystems can be an updater. All updates will be written directly to DASD by the OTHREADs. Subsequent read-only systems will see the updated DASD, but with the risk of seeing partial updates or getting GG status codes. The original read-only system will be using a data space. If all

CIs had been read in before the updating subsystem made any updates, the first system will have a clean point-in-time copy of the database, which gets progressively more and more out of date. If the updates started before all of the CIs had been read into the data space, then the VSO copy will be “fuzzy,” may contain partial updates, and may cause GG status codes that will remain no matter how many times the application retries the DL/1 call.

Logging and /ERE

To expedite emergency restart, some changes have been made to the use of log records for VSO areas (see Table 2).

Table 2. VSO Log Records

Log Record	***VSO Usage***
5612	Used to indicate output thread completion for a unit of recovery. However, VSO updates are not included in this UOR. In other words, the 5612 log record indicates completion of non-VSO write I/Os.
5910 (new)	For a VSO area, indicates the start of writing updated CIs from the data space to DASD, either by periodic write-back or system checkpoint.
5912 (new)	Indicates the end of the VSO write-backs for an area.

At emergency restart, a temporary data space is obtained if there are any VSO areas to be recovered. Reading the log from the restart checkpoint, application updates (5959 log records) of VSO CIs are tracked in this data space. When the end of log is encountered, any VSO updates following the last 5912 log record may or may not have been written out to DASD. Fast Path reads each CI from DASD into the data space, applies the updates, and writes it back to DASD. At the end of /ERE processing, the temporary data space is released.

Note: It is likely that /ERE will have more I/Os to do when VSO is used.

I/O Errors with VSO

If a write error occurs during write-back, an Error Queue Element (EQE) is built, as usual. However, the data is still in the data space and is available for subsequent use by applications. No further attempts will be made to write this CI. The appropriate recovery utility may well be the Area Create Utility, especially if all of the CIs are already in the data space. All the SDEP CIs must be readable as well.

A read I/O error during VSO Preload will cause that CI to be skipped. At the first application request for that CI, the read will be attempted again. If it also fails, the application will receive an AO status code, and all subsequent requests will immediately be given an AO status code. No further retries of the read I/O will be attempted.

In the case of an I/O error on the first read request by an application program (that is, non-preload or a request before preload completes), the application receives an AO status code and subsequent access to the CI will not be possible.

Operator Commands

VSO specifications are defined in DBRC and can be changed at any time.

The /START AREA command is used to cause VSO options to be invoked. It is not necessary to stop the area first. The command only affects an area that is not in virtual storage. For example, if a non-VSO area is in use, you can define it to DBRC as VSO and PRELOAD, issue the /START command, and Fast Path will load the area into a data space. However, a /START command after changing NOPREL to PRELOAD would not move an area from the non-DREF data space to the DREF data space. Further, /START AREA cannot be used to switch an area from VSO to non-VSO. A new command is provided for this purpose, namely /VUNLOAD.

The /VUNLOAD AREA command unloads an area from the data space. All updated CIs are copied to DASD and the storage is released. To do this and maintain full application accessibility would be very complex. So the Fast Path Sync Latch is held for the duration. This allows applications to read from the data space, but when they enter syncpoint, they will be forced to wait on the latch. When the unload completes, the latch is released, and syncpoint processing will continue. All the pending updates are written out to DASD.

Consequently, the /VUNLOAD AREA command should be used with caution. If the target area is updated only occasionally, there would be no problem. If the target area is quite large and very heavily updated, transaction processing might stop for seconds or even tens of seconds. It is probably a good idea to issue a simple checkpoint command before the /VUN command to initiate the standard write-back processing. This will minimize the number of new updates that have to be flushed out by the unload processing.

The /VUN command works regardless of the DBRC current VSO specification. It is not remembered across IMS restarts. Whenever IMS is restarted, the DBRC specifications will always be honored. Similarly, /STA AREA always interacts with DBRC to establish the current options (with the exception of changing from VSO to NOVSO).

The /STOP and /DBRECOVER AREA commands cause the area to become unavailable, outstanding updates are copied from the data space to DASD, the data sets are closed and deallocated, and the data space storage is released.

The /DISPLAY command has a new option, FPVIRTUAL, which shows which areas use VSO, what their VSO options are, which data spaces there are, and how much of each data space is in use.

The /START DB ACCESS= command should be used with caution. It causes all areas to be closed (Access Intent of a DEDB cannot be changed while any of its areas are open). In the case of VSO areas, pending updates in the data space will first be written back to DASD and then the areas will be removed from the data space. At the next program access request, the area will be reopened. However, for a VSO area, VSO options will not be implemented. A /STA AREA command is required to invoke the VSO functions.

VSO in an XRF Environment

The alternate system will have one more data space than the active system. The alternate system maintains standard VSO data spaces to match the VSO data spaces on the active system, but these are not used in tracking mode. The alternate also has a non-DREF data space to keep track of the active system′s VSO updates, so that in the event of a takeover, it can perform the necessary /ERE-type processing. Failure to acquire this tracking data space would result in any updated VSO DEDBs being stopped and flagged as “recovery-needed” on the alternate system after takeover.

The alternate system tracks operator commands that relate to VSO. After takeover, the VSO options that were in use on the old Active system will be in use on the new active system.

Note: This is different from an /ERE, where VSO areas are always started with the current DBRC options.

The standard VSO data spaces on the alternate system only begin to get used after a takeover. Consequently, for areas using PRELOAD, they get preloaded at the end of takeover. Any application requests for CIs that have not yet been preloaded will be satisfied from DASD.

3.2.2 Implementing VSO

The implementation of VSO is very easy. The tasks discussed below are required for the definition and activation of a DEDB using VSO.

DBRC

All VSO DEDBs and areas must be registered with DBRC, specifying the required VSO options. The appropriate commands are:

```
CHANGE.DBDS options
INIT.DBDS options
```

The relevant options are:

```
VSO or NOVSO (default)
PRELOAD or NOPREL (default)
PREOPEN or NOPREO (default)
```

The PREOPEN or NOPREO option applies to all DEDB areas, VSO and non-VSO. (Additional details can be found in 3.2.7, “PREOPEN Attribute” on page 34.) The PRELOAD option incorporates the PREOPEN option. A NOPREL VSO area can have either PREOPEN or NOPREO specified. If you expect to switch between PRELOAD and NOPREL but always want PREOPEN, specify both options.

Starting to Use VSO

Once the DBRC definitions are complete, it is simple to start using VSO. Either restart IMS or /START each VSO area. In the event of a need to fall back to non-VSO, issue the /VUNLOAD command, and update DBRC with a CHANGE.DBDS NOVSO command.

Other Implementation Considerations: There must be sufficient real storage (central and expanded) to support the VSO data spaces. Performance of non-preloaded VSO areas in particular could be severely impacted if real storage is insufficient.

The maximum size of an area data set is 4GB, whereas the maximum size of a VSO area is a little under 2GB. If VSO is to be used for an area greater than 2GB in size, that area must first be split into two or more smaller areas. In IMS 5.1, no facilities exist to simplify this task. Changes to the DEDB randomizer and a complete batch unload/reload of the whole database are required.

The new version of the Fast Path Log Analysis Utility (described in 3.2.14, “Fast Path Log Analysis Utility” on page 43) includes VSO statistics.

3.2.3 Migration Considerations for VSO

VSO does not inhibit fallback from IMS 5.1 to a previous release. Consequently it is possible to start using VSO as soon as IMS 5.1 is put into production. Despite the relative ease of moving to and from VSO, many IMS customers have a very strict change control policy that prohibits any change at other than prescribed points in time. Such customers, implementing VSO as an option for existing areas, might choose to do all testing with VSO and then implement the live production system with VSO already in place.

Customers planning to convert MSDBs to VSO DEDBs would probably choose to implement IMS 5.1 with MSDBs and migrate to VSO DEDBs at a later date. A migration utility is provided to convert between MSDB and VSO (and vice versa), and it would help in this later phase of IMS 5.1 implementation. This utility and migration in general from MSDBs to DEDBs are covered in 3.2.5, “MSDB to DEDB Conversion” on page 27.

3.2.4 Value of VSO

VSO provides high performance and high concurrency of data access on top of the existing DEDB capabilities of high data availability and the unique DEDB functionality. VSO is particularly appropriate for use in table or control databases that are accessed by many transactions.

I/O Reduction

The most immediately obvious benefit from VSO is a reduction in I/Os. Some customers have areas that, without VSO, experience very high I/O rates, several 100 per second in some cases! Such a high I/O rate is made possible only by exploiting DASD controller facilities such as caching; a situation that is possible but generally not desirable, as it dominates the DASD controller′s resource and can have a serious impact on the overall I/O subsystem. Implementing such areas with VSO will provide major relief. The read I/Os will be eliminated, and the write I/Os will be optimized over time rather than done for one transaction at a time. Small, highly volatile databases will especially benefit from the deferred write I/Os, as updates to a CI from multiple transactions will be applied with a single I/O.

Some customers may not have any one area that is exceptionally busy, but the aggregate I/O rate against the database as a whole may stress the I/O subsystem. Again VSO, given sufficient central and expanded storage, might provide cost-effective relief.

The value of VSO in reducing I/Os is already well proven. The IBM TCP-A benchmark for MVS, published in December 1993, used an IMS Fast Path system that held all of its data in VSO areas.

Reducing Lock Contention

Non-VSO DEDBs never have database updates applied before syncpoint. Consequently, there is never a need for database backout. This leads to shorter pathlengths and reduced logging (no “before images”). But another consequence is that the CI lock has to be held until after the CI has been written out (it would be impossible without backout to handle the case where transaction 1 has its updates committed and transaction 2 abends after updating the same buffer). Again for performance reasons, Fast Path does not write out CIs to DASD immediately. Consequently the CI lock is held for a relatively long time (though rarely more than half a second).

For most transaction driven systems, typically with large databases, this locking is not a problem. The occasional CI contention will pass unnoticed. However, with small areas, even transaction systems can suffer from locking contentions. Where it is more common is in BMPs that perhaps do not use the GC status codes to drive syncpoint processing. After syncpoint, the BMP immediately asks for the previously processed CI again and has to wait on the lock while the CI is written out (which might be at the end of a long chain of I/Os). In this way some BMPs can spend a significant proportion of their elapsed time waiting on CI locks.

With VSO areas, the lock is released as soon as the update has been copied to the data space in syncpoint phase 2, effectively eliminating CI contention. Even if an area is thought to be too big to reside permanently in a data space, it is possible to assist BMP processing by updating DBRC with the VSO PRELOAD option, issuing /START AREA, letting the area get loaded into the data space, running the BMP (which should now go like lightning—no I/O and no contentions), issuing /STOP or /VUN AREA, and resetting the DBRC option to NOVSO.

Replacing MSDBs with VSO DEDBs

Another major reason for using VSO is as a replacement for MSDB[1]. MSDBs have often been used because of their exceptional performance. They have always had some restrictions, but ones which customers could accept. These are single segment hierarchy, no variable length segments, no ISRT or DLET calls allowed (in general), no DBRC support, and nonstandard backup/recovery utilities and procedures.

However, a more serious restriction became apparent with the advent of APPC/IMS. Applications that update MSDBs cannot be invoked by an LU6.2 “terminal.” Customers with MSDB updating applications have been unable to exploit APPC/IMS. Such customers can, with IMS 5.1, convert their MSDBs to VSO DEDBs, achieving comparable performance, and can then begin to implement APPC/IMS without restriction. Certain facilities for MSDBs have previously been unique to MSDBs. However, these facilities are now available for VSO DEDBs (and in general for non-VSO DEDBs as well), so enabling a straightforward migration without the need for application program changes in most cases. More details can be found in 3.2.5, “MSDB to DEDB Conversion” on page 27.

[1] MSDBs with LTERM and user-defined keys are still available in IMS 5.1, but no further enhancements are planned.

Some CICS DBCTL users have wanted access to MSDBs. This is not possible. But VSO DEDBs provide an even better alternative and are available in a DBCTL subsystem.

In summary, some of the benefits of VSO DEDBs compared with MSDBs are:

- Full DL/1 hierarchy
- Fixed or variable length segments
- Updatable by LU6.2 terminals
- Available to DBCTL systems
- Allows ISRT and DLET
- Includes full DBRC support
- Uses standard backup/recovery procedures and utilities.

A fuller summary can be found in Table 3 on page 30.

3.2.5 MSDB to DEDB Conversion

Given the benefits summarized in the previous section, many users of MSDBs will want to convert them to VSO DEDBs. To make this possible, those facilities that were unique to MSDBs have been implemented for DEDBs. Mostly, these new DEDB facilities apply equally to VSO and non-VSO areas. The new DEDB facilities are:

- Fixed length segments
- DL/1 Field (FLD) call
- DEDB PCB option: VIEW=MSDB
- Segment level locking (for restricted subset of VSO only)
- DBD FIELD TYPE=H|F.

In addition, a new utility is provided to aid in the conversion of MSDBs to DEDBs (and vice versa).

Fixed Length Segments

This is a new DBD option for a DEDB. Previously, only variable length segments were allowed for DEDB. For example:

```
SEGM NAME=ABCDEFGH,PARENT=0,BYTES=(400,20)
```

IMS 5.1 now allows fixed length segments. For example:

```
SEGM NAME=ZYXWVUTS,PARENT=0,BYTES=(220)
```

The LL bytes still exist in a fixed length segment (immediately before the user data), but they are not visible to application programs.

Segment compression and expansion routines treat all DEDB segments as variable length.

DL/1 Field (FLD) Call

The purpose of the FLD call is to update a segment in such a way that a non-shared lock is taken only at syncpoint time and held for a short duration. This locking technique enables the same data to be accessed and updated by multiple application programs executing in parallel and is ideally suited for use with very small but heavily updated databases. In the past, this is just what MSDBs have been used for. Now in IMS 5.1, DEDBs with the VSO option are likely to be used in the same way.

The technique is possible because application programs, using the FLD call, never actually see the values of fields in the segment. With the FLD call, the program specifies any number of tests (including none) to be performed on fields within the segment identified in the DL/1 call's segment search argument (SSA). These are called *verifies*. The FLD call also typically includes one or more changes to be made to fields in the segment (set a value, add or subtract a value). These changes are made only if all verifies are satisfied. When the program issues the FLD call, all verifies are executed. If any fails, a nonblank PCB status code is returned. If all the verifies are satisfied, Fast Path remembers both the verifies and the changes, but no attempt is made to do the changes. At syncpoint time, the segment is locked, and the verifies are reexecuted. If all are still satisfied, the changes are executed. If any reverify fails (other programs could have updated the segment since the FLD call was performed), the syncpoint is aborted, and the transaction is rescheduled (or a BMP would get a nonblank status code to its SYNC or CHKP call). When the transaction is reprocessed, the FLD call will experience a verify failure at call time, and the application will do the appropriate processing. Further details on the DL/1 FLD call can be found in *Application Programming: Database Manager (SC26-8015)*.

Implications of FLD Call with DEDB

At DL/1 call time, a FLD call against an MSDB takes a lock on the segment, just for the duration of the FLD call. In the case of a DEDB, no lock is taken at FLD call time. The CI is either already available to the program in a Fast Path buffer or read from DASD as if with PROCOPT=GO.

At syncpoint time the resource is locked and, assuming that all reverifies are satisfied, the changes are made to the data. For an MSDB, these changes are made directly to the database, and then the lock is released. For a DEDB, the updates are made to the CI in a Fast Path buffer. For a VSO area, these updates get copied to the data space as part of syncpoint processing, and the lock on the data is released. So, the FLD call works for a VSO area in just the same way as for an MSDB. For a non-VSO area, the updated CI is retained in the buffer, and the lock remains held until physical logging has occurred and the OTHREAD has completed its database write. So, much of the benefit of the FLD call is lost with non-VSO areas.

There is a further consideration concerning the use of the FLD call on a variable length DEDB segment. If, at syncpoint time, the DEDB segment is not in the same place as at the time of the FLD call, the reverify will fail. DEDB segments can move whenever a REPL call changes the segment length. Clearly this will not be an issue if fixed length segments are used, which may well be the case when the DEDB is a converted MSDB.

In terms of database positioning, the FLD call functions like a get unique (GU) call. Note that the target segment of a FLD call can be any segment in the hierarchy.

PCB Parameter: VIEW=MSDB

For an MSDB, all retrieval calls (for example, GU, GN, FLD) read the segment directly from the database. Updates are held in buffers until syncpoint time. Consequently, if a transaction retrieves an MSDB segment, updates it, and then retrieves it again, the second retrieval still gets the original value of the segment. Only at syncpoint are the updates reflected onto the database.

For application compatibility reasons, this unusual facility can be requested for a DEDB. If a program relies on it, the appropriate PCB should be changed to include VIEW=MSDB. For example:

```
PCB  TYPE=DB,NAME=MS2DEDB,PROCOPT=GR,KEYLEN=24,VIEW=MSDB
```

Fast Path handles this definition option by keeping an extra copy of each CI in the Fast Path Common Pool. These buffers are taken from the application region's normal buffer allocation (NBA). This is also true if HSSP private buffers are used for sequential processing against the database.

Several PCBs can be for the same database. In this case, some PCBs may include VIEW=MSDB, and others not.

Note: VIEW=MSDB is not a valid parameter for a PCB against an MSDB.

Segment Level Locking

MSDB locking is done at the segment level, whereas before IMS 5.1, DEDB locking was always done at the CI level.

In general in IMS 5.1, all DEDB locking for both VSO and non-VSO is still done at the CI level. However, for compatibility reasons, segment level locking has been implemented for any VSO DEDB whose characteristics exactly match those of an MSDB. In other words, segment level locking will be used (not an option) when the following conditions are met:

- DEDB has a root-only hierarchy
- The root is defined as a fixed length segment
- PCB PROCOPT=G or R (that is, no ISRT or DLET allowed)
- Area is a VSO area.

Whenever a segment lock is taken, there will also be a shared lock on the CI containing the segment. This is necessary to avoid problems if a program with PROCOPT=A, for example, is also accessing that CI (not using segment level locking). As long as all programs have a PROCOPT of G or R, the potential benefits of segment level locking will be fully realized.

The standard Fast Path locking facility is employed for locking, using XCRBs in as many IPAGEs as necessary. When there is contention for a segment, control is passed to the appropriate IMS lock manager (program isolation or IRLM).

It has also been necessary for Fast Path to implement a form of dynamic backout. Two or more programs can, with segment level locking, update different segments in the same CI. Should one of these programs abort rather than commit, it is not possible to simply discard its updates. In this case, Fast Path reads the original value of the segment from the data space and copies it into the buffer, before releasing the lock on the segment.

DBD FIELD TYPE=H|F

DEDBs now support two additional field types in the DBD that were previously only available for MSDB:

- TYPE=H → half-word binary field
- TYPE=F → full-word binary field.

MSDB and DEDB Comparison

Table 3 summarizes the key differences between MSDBs, VSO DEDBs, and non-VSO DEDBs.

Table 3. MSDBs and DEDBs

Function	MSDB	DEDB		
		IMS 5.1 VSO	IMS 5.1 non-VSO	Prior to IMS 5.1
Memory database capability	Yes	Yes	No	No
Multiple segment hierarchy	No	Yes	Yes	Yes
DBRC support	No	Yes	Yes	Yes
SDEP support	No	Yes•	Yes	Yes
APPC/IMS support	No	Yes	Yes	Yes
CICS/DBCTL support	No	Yes	Yes	Yes
Block level data sharing	No	No	Yes	Yes
Area level data sharing	No	Yes•	Yes•	Yes•
Standard DB recovery	No	Yes	Yes	Yes
Fixed length segment	Yes	Yes	Yes	No
Variable length segment	No	Yes	Yes	Yes
FLD call support	Yes	Yes	Yes•	No
Q command code support	No	Yes	Yes	No
ISRT or DLET calls	No•	Yes	Yes	Yes
Locking level	segment	segment• or CI	CI	CI

Notes:
1. SDEPs are not in data space
2. With restrictions
3. Not if DEDB has SDEP defined
4. Limited benefit
5. Not allowed for non-terminal-related MSDBs
6. See “Segment Level Locking” on page 29 for details of when segment level locking applies.

MSDB to DEDB Conversion Utility

The new MSDB to DEDB Conversion Utility, DBFUCDB0, simplifies the process of converting an MSDB to a DEDB. It is used in conjunction with other existing utilities, as part of an overall conversion procedure.

An MSDB is converted into a single area DEDB. The utility also handles fallback from DEDB to MSDB.

It is not possible to remove MSDBs online. Consequently, the replacement of one or more MSDBs with DEDBs is an offline batch process.

MSDBs exist outside IMS in two forms: a checkpoint data set or an MSDBINIT data set. Existing offline MSDB updating is done with the MSDB Maintenance Utility, which processes an MSDBINIT data set. So for convenience, the new MSDB to DEDB Conversion Utility also processes an MSDBINIT data set. (When running in fallback mode, it creates an MSDBINIT data set.)

Two control statements are used to determine the function of the MSDB to DEDB Conversion Utility:

- TYPE statement

  This statement can contain either of two values, depending on whether an MSDB is being converted to a DEDB, or a DEDB is falling back to being an MSDB:

  - TYPE=CONVERT
  - TYPE=FALLBACK.

- Database statement

  This statement specifies the names of the MSDB and DEDB involved:

  ```
  MSDB=XXXXXXXX,DEDB=YYYYYYYY
  ```

  Only one MSDB can be processed in one execution of the utility.

Forward Conversion (MSDB to DEDB)

As always, the MSDBINIT data set is created after IMS closedown using the MSDB Dump Recovery Utility, which reads the appropriate MSDB checkpoint data set and reformats it into an MSDBINIT data set. The MSDBINIT data set is then provided as input to the MSDB to DEDB Conversion Utility.

It is not necessary to edit the MSDBINIT data set to remove MSDBs that are not being converted, as the new utility selects only the databases specified on control statements. Similarly, at the eventual IMS restart, IMS will just select the MSDBs defined in the specified DBFMSDBx member.

The new utility does not directly load the data into the area data set. That is done with a batch reload utility. IBM supplies such a utility in IMS Database Tools, a separate Program Product (5685-093). The appropriate tool is the DEDB Unload/Reload Utility, commonly used to perform offline DEDB area reorganization. This reload, or an equivalent utility, is used to format and load an area with the data from the MSDB. The MSDB to DEDB Conversion Utility will reference the ACBLIB containing the DEDB DBD to ascertain the area name. The ddname in the JCL for the output data set must be this area name.

Fallback Conversion (DEDB to MSDB)

The MSDB to DEDB Conversion Utility, running in fallback mode, reads in a DEDB batch unload file. This file must be created by first running the unload supplied in the Data Base Tools DEDB Unload/Reload Utility, or an equivalent product. Next, the unload output data set is sorted into key sequence.

The MSDB to DEDB Conversion Utility is then run in Fallback mode to convert the sorted DEDB unload file into an MSDBINIT data set. The input ddname must equal the area name being converted.

The converted output is a single MSDB in MSDBINIT format. The MSDB Maintenance Utility should be run to merge this MSDB with the online system′s MSDBINIT data set, ready for IMS restart.

Customized Conversion

The actual record conversion is done by an exit routine: DBFUCDX0 in convert mode and DBFUCDX1 in fallback mode. The user can modify an exit and relink DBFUCDB0 to include the revised exit routine. This might be appropriate, for example, if an alternative to the IBM DBT DEDB Unload/Reload Utility is to be used.

3.2.6 MSDB to DEDB Migration Considerations

The migration process can be smooth if it is well planned and carefully executed.

Conversion Procedure

The physical conversion of an MSDB to a DEDB is significantly eased by the DBFUCDB0 utility described in “MSDB to DEDB Conversion Utility” on page 30. But this is just one step in the required procedure, which, as usual, needs careful planning and testing.

There are two approaches to the conversion. Each PCB contains the DBD name of the database. So either the same name is used for the DEDB and the MSDB, in which case the PSBs may not need to be changed, or the MSDB and the DEDB are given different names, in which case each application will require a new program specification block (PSB) with changed PCBs. (If you use VIEW=MSDB, the latter approach will have to be the case).

Either way, two ACBLIBs will be needed, one containing the MSDB DBD and the other the DEDB DBD. The PSBs will be in both ACBLIBs, and either the unchanged PSBs are in both or the original PSBs are with the MSDB DBD and the revised PSBs are with the DEDB DBD.

The MSDB to DEDB conversion steps are:

1. Register new DEDBs with DBRC, specifying appropriate VSO options.
2. Create the DEDB DBD.

   Note: It is the user′s responsibility to ensure that the field names in the MSDB and DEDB segments are the same.
3. Create new PSBs if necessary.
4. ACBGEN the new DBD and PSBs into the new ACBLIB.
5. Run the MSDB Dump/Recovery Utility in UNLOAD mode to create the MSDBINIT data set.
6. Run the new MSDB Conversion Utility with the CONVERT option to create reload data sets for each MSDB being converted.
7. Run the batch Reload Utility from DBTools (or an equivalent product) to load the areas.

IMS will need to be restarted with the new ACBLIB. If all MSDBs have been converted, the IMS Control Region JCL can be tidied up (remove the MSDB= parameter, MSDB checkpoint, and MSDBINIT data sets). If not all MSDBs have been removed, create a new DBFMSDBx member and change MSDB=suffix in the Control Region JCL. The original MSDBINIT data set will still be appropriate—only the remaining MSDBs, as specified in DBFMSDBx, will be selected.

Ensure the old ACBLIB is retained for fallback purposes.

Note: A similar procedure will need to be written and tested for use in the event that fallback is needed.

Segment Length

An MSDB segment length can be upto 32000 bytes. The maximum size DEDB segment is 120 bytes less than the CI size. Typically DEDB CI sizes are 4KB or 8KB. So there is clearly a problem for any customer with an MSDB segment length greater than the maximum DEDB CI size. Using a larger CI size than normal is possible but unattractive because all common Fast Path buffers must be of this size (or bigger).

Probably the best solution is to remain with MSDBs until the database has been redesigned to use multiple segments per database record, and the applications have been modified to use the new hierarchy.

Segment Sequence

A significant difference between an MSDB and a DEDB is the physical order of the segments. In an MSDB, the segments are in key sequence. In a DEDB, the order is determined by the randomizer and in general is not the same as the key sequence. If applications only use SSAs with “key=value,” there is no problem. However, applications with SSAs like “key>value” will have different results for MSDBs and DEDBs. The only solution in this case is to implement a sequential randomizer for the DEDB.

VIEW=MSDB

It will be necessary to assess the value of the PCB VIEW=MSDB option (see “PCB Parameter: VIEW=MSDB” on page 28). It is possible that some applications may inadvertently be exploiting the facility with MSDBs today. It may even be that a programmer may have deliberately exploited the facility, in which case there may or may not be some reference to it in the program documentation. Consequently, it is difficult to be 100% sure of which programs, if any, exploit the MSDB facility.

In migrating to DEDB, then, the safest way is to include VIEW=MSDB in all appropriate DEDB PCBs. However, this probably is not necessary or desirable. If there is no known deliberate exploitation of the facility, standard application regression testing should highlight any program dependencies.

Buffer Allocations (NBA)

It may be necessary to increase the dependent region NBA sizes. Applications might use more buffers when accessing DEDBs that have been converted from MSDBs for two reasons:

- MSDB segment retrievals do not use Fast Path buffers. Consequently, applications that do not update MSDBs do not use Fast Path buffers. When an MSDB is converted to DEDB, buffers are required for data retrieval.
- If the VIEW=MSDB option is used, Fast Path requires additional buffers in which to keep the original values of the data.

Locking

MSDBs always lock at the segment level. If all applications continue to use PROCOPT=G or R and no changes are made to the DBD (other than to change it from an MSDB to DEDB), segment level locking will continue to be used. In this case there should not be any increased contention.

If changes are made that inhibit segment level locking (see “Segment Level Locking” on page 29), CI level locking will be used, but the CI locks (for VSO CIs) will be released at syncpoint time. Nevertheless, this can still cause problems for high volume transactions with large numbers of updates (REPL, ISRT, DLET rather than FLD) to small databases. In this case, the recommended solution is to implement one root segment per CI, perhaps with a smaller CI size (for example, 512 bytes). Note that this may require that a larger NBA be defined for the relevant applications, because a buffer can only hold one CI, and hence each buffer will hold less data than before.

CPU Cost

There is always a CPU cost inherent in migrating from a low function to a high function facility. In the case of MSDB to VSO DEDB migration, these costs are modest but clearly dependent on the application profiles. As a rough guide, one might reasonably assume that a DL/1 call path length will increase by 1.5K when moving from an MSDB to a VSO DEDB. For an application with 10 MSDB accesses and a total pathlength of 200KI, the increase in CPU pathlength would typically be about 7.5%.

Logging

MSDB updates are logged with 5920 log records, whereas DEDB updates use 5950 records, which are longer. For most customers, this difference will go unnoticed. A customer with a proportionally large volume of MSDB updates might see a small percentage increase in application log data.

3.2.7 PREOPEN Attribute

Some users like to have some or all of the areas opened and available before the application programs begin processing to avoid early transactions being delayed by the VSAM data set opens. Before IMS 5.1 this could be done only by writing special programs, submitted by an automated operator at IMS startup, that accessed all available areas. An unqualified DL/1 POS call was a useful tool in doing this.

With IMS 5.1, it is now easy to implement such a requirement, using the PREOPEN facility. Areas needing to be preopened are defined to DBRC with the PREOPEN attribute using CHANGE.DBDS (or INIT.DBDS at Area Registration). For example:

```
CHANGE.DBDS PREOPEN
INIT.DBDS PREOPEN
```

The options are:

```
PREOPEN or NOPREO (default)
```

The NOPREO option is needed to enable the PREOPEN attribute to be turned off.

Whenever IMS starts or /START AREA is used to make an area available again, DBRC informs Fast Path of any areas with the PREOPEN option, and they are

allocated and opened immediately. For VSO, PREOPEN is assumed when PRELOAD is specified.

3.2.8 PROCOPT=GO

Before IMS 5.1, PCBs could include PROCOPT=GO. However, if the IMS subsystem had an access intent of other than read-only (RO), Fast Path would upgrade the PROCOPT to G and take CI locks while reading the database.

In IMS 5.1, PROCOPT=GO is honored as a request to read without locking. PROCOPTs of GON and GOT are also accepted. Should an invalid pointer be detected, Fast Path takes the following actions:

- GO results in a 1026 application abend.
- GON results in a GG status code.
- GOT results in a GG status code.

Pointer errors can occur if the PROCOPT=GOx program reads a CI containing a pointer into another CI. Before that second CI is read, however, an OTHREAD causes it to be updated, perhaps deleting the target segment.

On getting a GG status code, the application should not attempt to retry the access. The invalid pointer is not going to change—it is in the buffer owned by the PROCOPT=GOx program. It is for this same reason that Fast Path, unlike Full Function DL/1, does not attempt a retry with PROCOPT=GOT when it detects a pointer error.

CIs accessed through PROCOPT=GOx will never contain uncommitted updates because PROCOPT=GO buffers are managed separately from the rest. In satisfying a PROCOPT=GO request, Fast Path looks only at which buffers have been used by the program for previous PROCOPT=GO requests. If the required CI is not found, it is read in from DASD. So, for example, consider a program with two PCBs for the same DEDB, one with an update processing option and the other with PROCOPT=GOx. If both PCBs happen to access the same CI, there will be two copies of the CI in the buffer pool. The PROCOPT=GOx buffer will remain unchanged. The other will contain uncommitted updates until syncpoint time.

SDEP CIs are also read without locking. After an SDEP CI has been read into a buffer, the next CI to be read will always reuse that same buffer. However, even though PROCOPT=GOx means there is no locking, the SDEP CI buffering scheme is not used. Buffers used for reading with PROCOPT=GOx are only released at syncpoint, ROLB, or buffer steal time. However, if buffer steal finds a buffer used by PROCOPT=GOx, it reclaims it and does no further stealing.

3.2.9 Q Command Code

It is sometimes an application requirement that the processing be determined by a set of data values meeting some criteria. For example, only dispatch an order if all parts ordered are available in the warehouse. If the determining data is spread across multiple segments, the application needs to lock each segment after it has read it, to stop other programs from updating it before the full set of criteria has been evaluated. With Full Function DL/1, this locking is possible with the Q command code. But with DEDB, this has been possible only if the application updated each segment, because if buffer steal is invoked, it can release buffers that have not been updated.

IMS 5.1 addresses this problem by allowing the Q command code to be used with DEDB.

The Q command code is valid only on get calls, and it causes the buffer containing the retrieved segment to be flagged as not stealable by buffer steal. It does not cause any additional locks to be taken.

The standard DL/1 syntax must be followed:

- It is a two-byte command code.
- The first character is Q.
- The second character is a letter between A and J.

However, the second character (the ENQ class for DL/1) is ignored. The PSB MAXQ parameter is also ignored—there is no limit on the number of DEDB get calls allowed with the Q command code. The Q command code works at the CI or segment level, as appropriate.

3.2.10 DEDB DEQ Call

IMS 5.1 introduces the DL/1 DEQ call for DEDBs to:

- Release the data that the application has locked with the Q command code
- Enable an application to invoke buffer stealing when it suits the application, rather than when all of the application's buffers become full.

The DEDB DEQ call must be issued against a DEDB PCB—it does not make any difference which DEDB PCB is used. This is different from the Full Function DL/1 call, which is issued against the IOPCB. The two flavors of DEQ call can be used in the same program, one acting on full function databases, the other on the DEDBs. Unlike for full function, the DEDB DEQ command does not specify the class of lock to be dequeued.

With the DEDB DEQ call, any buffers flagged by the Q command code are unflagged, and buffer stealing is invoked.

Application Invocation of Buffer Stealing

There are cases where it is extremely beneficial for the application to be able to invoke buffer stealing. Consider a complex enquiry transaction that typically reads multiple database records. Each root is retrieved with a GU (GN ROOT processing would not be appropriate for this example). Each database record is large and spans multiple overflow CIs. It would be impractical for the region NBA to be large enough to hold all of the CIs referenced. Instead, once all of the NBA buffers have been filled and another buffer is needed, Fast Path will invoke buffer steal. Probably, the application will be part way through processing a database record when this happens, and it is likely that some buffers will be released that contain segments that have not yet been processed. When the program moves on to request these segments, CIs will have to be read in for a second time. This transaction already has a high I/O requirement, so it is most undesirable to have to read some CIs multiple times!

Using IMS 5.1, the solution is simple. After each root segment retrieval (other than the first), issue the DEQ call. This will release all of the buffers containing CIs for the previous database record and so provide the maximum possible number of buffers for the current database record.

Note: Buffer steal will not steal the CI containing the current root for any PCB.

If no buffers are reclaimed, the DEQ call will receive an FW status code.

3.2.11 DEDB High Speed Reorganization Utility

The online reorganization utility for DEDB, DBFUMDR0, was previously called the DEDB Direct Reorganization Utility. It has been rewritten and given a new name that reflects one of its new characteristics, namely, DEDB **High Speed** Reorganization Utility.

The objectives of the new utility are twofold: To increase performance by exploiting data-in-memory techniques, reducing I/Os, and using asynchronous rather than synchronous I/Os and to reduce the amount of log data produced.

Function

The original utility uses the Reorg UOW as a work area on the disk. Each UOW is locked, and reorganized into the Reorg UOW. This requires I/Os to the Reorg UOW and typically IMS logs the whole CIs. Each CI from the Reorg UOW is then copied back to the source UOW, again requiring I/Os and full CI logging. In effect, the whole of the direct part of the area is read, written back, and written to the log twice!

The new DEDB High Speed Reorganization Utility has some major differences. It does not use the Reorg UOW. Instead it runs just like an HSSP BMP, reading the CIs into private buffers, updating them, invoking syncpoint processing, logging the segment changes, and using an OTHREAD to do asynchronous write I/Os. This technique saves about half the original I/Os and logs less than half the original amount of log data.

If the user allocates sufficient buffers, the utility will also do asynchronous read-ahead of the next UOW while it is reorganizing the current UOW. This will be especially beneficial if the IOVF part is on a separate DASD volume, so that there is no device contention between the asynchronous process and the IOVF reads. The read-ahead requests a lock on the UOW. If the lock cannot be granted, the read-ahead is not attempted for that UOW.

Reorganization of a VSO area is exceptionally fast! It might be worth considering temporarily loading a non-VSO area into virtual storage, reorganizing it, and then unloading it again.

Note: For compatibility reasons (that is, fallback to a previous release) the Reorg UOW has not been removed, but it will not be used by IMS 5.1.

Implementation

The new utility runs just like the previous version, as a standard online Fast Path utility. From an externals point of view, there is no change.[2] However, the BUFNO parameter has a new meaning. It now specifies the number of sets of buffers that are to be initially allocated. The size of a buffer set is the UOW size for that area. The buffers are private buffers, and they are page fixed.

If necessary the pool of buffers can be dynamically extended up to a maximum of twice the BUFNO number of buffer sets. The default BUFNO is 3. This setting will disable asynchronous read-ahead. A BUFNO of 4 or more will cause look-ahead buffering to take place.

---

[2] Actually there are two new operators for the TYPE command: HSR and HSREORG. The original operator, REORG, is still valid.

Buffers are used to hold the UOW being processed, the reorganized UOW, and any IOVF CIs (hence the default BUFNO of 3). With BUFNO>3, one set of buffers is used for the read-ahead UOW. The private pool will be extended one buffer set at a time whenever a request for a buffer cannot be immediately satisfied, perhaps because there are a large number of IOVF CIs or the OTHREAD is slightly lagging behind. When appropriate, Log Check Write requests will be issued to force physical logging and so expedite OTHREAD scheduling. If the maximum number of buffer sets (2 x BUFNO) is not enough because of a very large number of IOVF CIs, that UOW will be skipped. This situation will be reported in SYSPRINT.

Because they are needed by the Fast Path system as a whole, the IOVF space maps are not read into private buffers; they use the Fast Path Common Pool instead to ensure that normal locking protocols are followed. It is not necessary to lock data CIs from IOVF, because they can be accessed only through the UOW that owns them, and the utility takes a lock on the UOW.

Migration Considerations

The change to using this new utility is straightforward, provided that the changes in real storage usage and restart processing are anticipated.

Real Storage Requirements: The new utility uses a lot more real storage than the original version. For example, an area with a CI size of 8KB and a UOW of 24 CIs running with BUFNO=4 would use a minimum of 768KB and a maximum of over 1.5 MB of page fixed buffers.

Consequently, customers might want to limit the number of Reorganization Utilities that are run in parallel. Concurrent HSSP BMPs should also be considered at the same time.

Restart Processing: In previous IMS releases, special processing had to be done at /START AREA or /ERE if a failure had previously occurred while the Reorganization Utility was copying the Reorg UOW back to the source UOW. This is no longer the case. The new utility is treated just like any other updating job. All recovery is standard, based on log records.

Value

Fast Path has always excelled in providing high data availability. A unique feature has been the ability to reorganize areas online. Customers typically do this at quiet periods of the night or on the weekend. Nevertheless, the speed of the DEDB Direct Reorganization Utility has given some customers cause for concern, impacting their job schedules and even leading some to prefer a short outage while they run a batch reorganization.

With IMS 5.1, such concerns are eliminated. The new High Speed Reorganization Utility performs several times faster than before and so completes in a fraction of the time it used to take.

The amount of log data produced (up to twice the size of the area itself) has been viewed as excessive. This has been a concern especially for customers shipping log data to a remote site through communications lines. The advent of Remote Site Recovery in IMS 5.1 adds weight to this concern.

The new Reorganization Utility will certainly log less than half of what it used to, because at worst it will log each CI once. But this is very much a “worst case

scenario." The actual amount of log data will depend on the degree of disorganization. Segments that do not get moved will not be logged. So, typically, the amount of log data will be much less than half of what it used to be. This will be of benefit generally in reducing log volumes and numbers of log archives required. It will also reduce both the size of the change accumulated log and the recovery time for an area and could expedite an emergency restart.

From a remote site contingency perspective, the new utility significantly reduces the amount of log data that needs to be transmitted by RSR or other online log shipping facilities.

3.2.12 High Speed Sequential Processing Enhancements

HSSP has undergone major redesign in IMS 5.1 to provide enhancements to both the basic processing and the Image Copy option. This section discusses the basic enhancements, and the section that follows considers the new Image Copy facility.

HSSP became available in IMS 3.1. It introduced the concepts of private buffers and UOW locking. But the main technique for speeding up processing was the exploitation of the sequential caching algorithm available on the 3990-3 DASD Control Unit. It automatically gained the benefits of other caching facilities such as DASD Fast Write.

The degree of benefit achieved by customers has been very variable, depending on factors such as UOW size, size of cache, and the workload of the DASD Control Unit. IMS 5.1 addresses this by providing a lot more performance-related function in Fast Path software.

The HSSP enhancement in IMS 5.1 provides higher HSSP performance for both read-only scans and updating BMPs and consistently achievable performance, without dependence on DASD hardware.

The main enhancements are:

- Exploiting data-in-memory
- Doing reads asynchronously rather than synchronously
- Providing optional look-ahead buffering.

The basic concepts remain unchanged:

- The SETR statement can control which areas get processed.
- The SETO statement is used to specify HSSP options.
- UOWs are processed one at a time.
- Locking is at the UOW level.
- Private buffers are used rather than NBA in Fast Path Common Pool.

The new HSSP uses more private buffers. It initially allocates three sets of private buffers (the size of a set is the UOW size for the area), and this number can be dynamically increased, one set at a time, up to a maximum of six sets. All CIs from sequentially accessed UOWs will use these private buffers. All IOVF and SDEP CIs will be read into buffers in the Fast Path Common Pool and are taken from the region′s NBA. As in the original HSSP, a Log Check Write is issued at syncpoint time to ensure that OTHREAD scheduling is not delayed.

A significant difference is that, in the new HSSP, the Root Addressable Part CIs are not read individually when requested by the application. Instead, the whole UOW is always read in with a single chained I/O.

By default, HSSP will do look-ahead reading of the next UOW. In cases where this is not desirable, such as skip sequential processing of clusters of roots, it can be turned off. A new SETO parameter, NORDAH, is used for this purpose.

Use of Cached DASD

HSSP will still request the sequential caching algorithm of a cached DASD controller when appropriate but clearly is not dependent on it. Chained reads and writes will perform better with caching, but HSSP does both asynchronously, and so the benefits may be marginal. The main benefit of sequential caching is perhaps to the cache itself, in that the HSSP will not monopolize great amounts of cache storage but will release used cache as fast as it acquires new cache.

Implementing HSSP

Implementing HSSP is largely unchanged from previous releases. The main requirement is to change a BMP′s sequential processing PCB PROCOPT from P to H (and use SETO NOPROCH to turn off HSSP again).

The only new feature from an externals perspective is the option to disable look-ahead reading. This is done with SETO NORDAH.

Migration Implications

The new HSSP function can be a candidate for implementation only if the following requirements are met:

Programming Restriction: A new programming restriction has been introduced with the new HSSP. HSSP has the concept of sequential position, which is set at the current UOW. Any requests for data in the current UOW are immediately satisfied from the private buffers. A request for data in a UOW beyond the current UOW causes a GC status code, a SYNC or CHKP must be taken, and the sequential position moves on to the next requested UOW. A request for data from a UOW before the current UOW is called a “backward reference.” In previous releases of HSSP, backward references were allowed. The program still got a GC status code, but sequential position did not move. (The enforced syncpoint at this time causes the current sequential position to move on to the next UOW). In IMS 5.1, backward references are not allowed. They receive an FY status code. Programs should be given an additional non-HSSP PCB to be used for backward references. Some customers, in implementing an earlier release of HSSP, modified their programs to have an extra PCB for “forward references” that were not part of the true sequential processing. This extra PCB would be ideal for backward references as well and may well already have been coded in this way.

Another implication of this is that the program must not take a syncpoint part way through processing a UOW. At syncpoint time, current sequential position moves on to at least the next UOW. So an attempt to continue processing the same UOW will be a backward reference and will receive an FY status code. This makes it essential to avoid syncpointing on FW status codes if possible, and the simple way to do this is to give the BMP a more than sufficient NBA. Should there be no way of avoiding FW, another PCB would need to be used, defined with PROCOPT=P, to continue processing after such a forced syncpoint and

until the next GC status code indicated that the end of the UOW had been reached.[3]

Note: A SYNC or CHKP does not have to be issued on getting an FW, but continuing with normal processing may risk running out of buffers, getting an FR status, and suffering an internal rollback.

Real Storage Requirements: Previous releases of HSSP used one or two buffer sets, page fixed in real storage (one buffer set if read-only, two buffer sets if updating). The new HSSP has a minimum of three buffer sets and a maximum of six. So a customer who runs 25 parallel HSSP BMPs against areas with CI size of 8K and UOW size of 24 CIs could require 25 x 6 x 24 x 8K = 28MB of real storage for the private buffers.

Planning for real storage usage should also take into account any concurrent DEDB High Speed Reorganization Utility runs.

3.2.13 HSSP Asynchronous Image Copy

IMS 5.1, as well as containing a redesigned HSSP, includes a completely new HSSP Image Copy option that replaces the previous option. It allows an image copy to be taken of an area while it is being processed by the HSSP application, and the copy can be to tape or DASD. The image copy produced is a standard Concurrent Image Copy.

The objectives of ASIC are to:

- Allow a standard image copy to be taken at the same time as an area is being processed by a sequential BMP, and so eliminate a subsequent sequential pass of the data set with an image copy utility.
- Improve the performance of the previous HSSP Image Copy.
- Remove the operational complexities of the previous HSSP Image Copy.

The original HSSP Image Copy facility was considered overly complex from an operational point of view. The main points of concern were that it:

- Only supported image copy to DASD
- Used a nonstandard image copy format
- Did not log database updates. HSSP IC was the only place where updates were recorded. Hence:
  - Dual copies (on DASD) were recommended
  - At least two generations were required (increasing DASD usage or requiring the use of DFHSM)
  - If HSSP IC terminated prematurely, the partial IC had to be treated as a log extension and could be required for database recovery or /ERE.

From a performance perspective, the original HSSP Image Copy implementation was synchronously writing to the image copy data set at syncpoint time (though using chained I/O). It had to, because it was effectively the log for the HSSP BMP.

---

[3] HSSP usage of private buffers makes FW less likely (assuming NBA is not reduced), but it is still possible if there are some very large database records spanning multiple overflow CIs. FW can also result from processing against a non-HSSP PCB.

The new HSSP Image Copy facility has been renamed the HSSP Asynchronous Image Copy facility, but the change is much more extensive than just using asynchronous I/Os. The main external characteristics are:

- The Image Copy is to tape or DASD.
- A standard image copy format (like Concurrent Image Copy) is used.
- DBRC dynamically allocates the IC data set from RECON information
- All updates are logged to the normal IMS log (as well as being captured on the HSSP Image Copy). This is essential for Remote Site Recovery, which must ship all update log records to the remote location.
- If the job terminates prematurely, the partially completed image copy is discarded.
- All recovery procedures are standard.

Internally, the HSSP Image Copy runs asynchronously from the HSSP BMP. In other words, as long as the image copy is keeping up with the processing, the BMP will not have to wait on the image copy.

Typically the image copy works one UOW behind the BMP because the updated UOW can be copied only after its changes have been committed. Hence at syncpoint the committed UOW is copied into another buffer set for the image copy to write out. This copying is necessary because otherwise the OTHREAD and the image copy would have to be synchronized in some way—the buffers could not be released until both users had finished with them.

As well as copying UOWs processed by the BMP, the image copy also has to write out any skipped UOWs, and finally the IOVF and SDEP parts of the area. It uses another set of buffers, reserved for this purpose (therefore the maximum number of buffer sets when ASIC is being used is seven and not six).

When this BMP skips one or more UOWs, it does not wait for the image copy to catch up and continues to copy committed UOWs. Meanwhile, the image copy will be reading the skipped UOWs from the area and writing them out to the image copy data set using its extra reserved set of buffers. If the BMP skips too many UOWs, it will eventually run out of buffers—all of the buffers in more than five buffer sets will be holding CIs that are either waiting for the OTHREAD or the image copy to write them out. In this situation, the BMP will have to wait until a complete buffer set is available for it to read the next UOW.

When the BMP reaches the end of its sequential process, the image copy still has the IOVF and SDEP parts to write out. The BMP does not wait for this either. It can terminate or, if appropriate, move on to another area of the DEDB. The image copy will continue reading and writing out the remaining parts of the area, one UOW at a time, until it is complete.

Implementation

BMPs to drive HSSP image copy must have a PCB for sequential processing with PROCOPT=H.

All DEDBs that use ASIC must be registered with DBRC. The areas must be registered with REUSE and GENMAX specified (this is not an option). For example:

```
INIT.DBDS DBD(abc) AREA(def) GENMAX(7) REUSE RECOVPD(7)
```

At least the GENMAX number of image copy data sets must be defined in DBRC for each area. They will be used in a round-robin fashion, the next one being selected each time HSSP ASIC is run.

The DFSCTL DD statement must be used in the BMP JCL to supply the SETO statement that controls the HSSP ASIC. For example:

```
SETO DB=dbname,PCB=pcbname,IC=(area names),2,2ABEND
```

This requests dual image copy of the specified areas, should they be processed by HSSP in this dependent region (and with matching DEDB name and PCB name). The last parameter, CONTINUE, 1ABEND or 2ABEND has a slightly different meaning from in previous releases. It now affects only what happens at BMP startup. CONTINUE means run the BMP even if all image copies fail in initialization. 1ABEND means run the BMP as long as one image copy can be initialized, and 2ABEND means run the BMP only if two image copies can be initialized.

Value

Most users of DEDB have some need to process areas sequentially (that is, physical sequentially using GN ROOT logic), and some users process each area multiple times per day in this way. Typically, these users also take daily image copies (some even more frequently) of all areas.

IMS 5.1 offers these users a simple way of combining the two sequential processes—all the complexity of the previous release has been removed. Some customers have several hundred areas, and so it is perhaps possible to remove that number of jobs from the daily batch (BMP) schedule. The schedule is completed more quickly, and thus the pressure on operations to complete the overnight work before the online day begins may be reduced.

3.2.14 Fast Path Log Analysis Utility

Despite their universal acceptance, online performance monitors do not provide the in-depth performance analysis available by postprocessing the IMS System log. The log analysis tool provided with IMS for analyzing all aspects of IMS Fast Path usage is the Fast Path Log Analysis Utility (DBFULTA0).

In IMS 5.1, DBFULTA0 has been significantly enhanced to provide:

- Transit time components in milliseconds
- Summary information on DL/1 calls, I/Os, VSO usage, buffer usage, and contention waits
- Optional detailed information on:
  - DL/1 call counts
  - Buffer usage
  - VSO usage.

The additional data available in the IMS 5.1 utility will make performance analysis and problem resolution easier, for both the overall system and individual programs.

The Fast Path Log Analysis Utility is fully described in the *IMS/ESA V5 Utilities Reference: System, SC26-8035*. Here, only the major enhancements are highlighted.

The utility functions in exactly the same way as in previous versions. The input is an IMS System log, and the output comprises a number of print reports (SYSPRINT) and two data sets (SYSUT1 and SYSUT2) containing transaction detail records. SYSUT1, the Total Traffic data set, contains a record for every transaction that was on the log within the time period specified in the utility control statements. SYSUT2, the Exception Traffic data set, contains a record for each transaction that exceeded transit time criteria specified in the utility control statements.

In IMS 5.1, the Total Traffic data set is essentially the same as in previous releases, although any user-written programs that process this file should be reassembled with the IMS 5.1 DSECTs. The Exception Traffic data set now has a record length of 252 bytes (used to be 143) because it contains the additional data that is optionally available on the printed Detail Listing of Exception Transactions.

Many customers set both SYSUT1 and SYSUT2 to DD DUMMY and use the utility just as a report generator, typically with no transit time selection criteria specified. (Every transaction is an exception!) Five reports are produced:

- Detail Listing of Exception Transactions (changed in IMS 5.1)
- Summary of Exception Detail by Transaction Code for IFP Regions
- Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs
- Summary of VSO Activity (new in IMS 5.1)
- Recapitulation of the Analysis.

Detail Listing of Exception Transactions: The page layout of this report has changed significantly to allow extra data to be printed. Two samples are included. Figure 5 on page 45 shows the report for a system running BMPs and the new High Speed Reorganization Utility. Figure 6 on page 46 is an extract from the analysis of a transaction processing system.

For each IFP transaction or BMP sync interval, there is one basic print line and up to three optional print lines. The basic print line contains the following fields that have been added or enhanced from previous versions (some fields have been dropped):

- Sequence number is seven digits (was five).
- PST number (region identifier) is included.
- Transit time components are in milliseconds (used to be tenths).
- Message lengths are no longer reported. The information is still in the records written to SYSUT2 and is summarized in the Summary of Exception Detail by Transaction Code for IFP Regions Report.
- Area data set I/Os are now broken down into reads and writes.
- VSO reads and updates are included.
- The number of buffers used is now an absolute number, not a value relative to NBA. (The number of OBA buffers used is reported in the optional buffer line).
- Counts of Contentions now includes Waits for the OBA Latch.
- Buffer Waits are now reported under the heading “Contentions.”

```
          LEGEND

          ┌──────────────────────────────────────────────────────────┐
          │                                                          │
          │   RT: REGION TYPE; I=IFP, M=MPP, B=BMP, D=DBCTL, U=UTILITY
          │   PT: PROCESS TYPE; H=HSSP, R=REORG                      │
          │   CONTENTIONS: CI; NO. OF WAITS FOR CI(S)                │
          │                UW; NO. OF WAITS FOR UOW (S)              │
          │                OB; NO. OF WAITS FOR OVERFLOW BUFFER LOCK  │
          │                BW; NO. OF WAITS FOR COMMON BUFFERS       │
          │   SF: SYNC FAILURE CODES - SEE UTILITY REFERENCE MANUAL  │
          │   BUF USE: TOTAL BUFFERS USED FROM THE COMMON POOL - INCLUDES
          │       NBA, OBA AND NRDB (NON-RELATED BUFFERS FOR SDEP/MSDB USE)
          │                                                          │
          └──────────────────────────────────────────────────────────┘


 SEQ  TRANCODE SYNC POINT  S ROUTING  LOGICAL  PST QUEUE TRANSIT TIMES(MSEC)- -OUT- DEDB ..ADS.. ..VSO.. MSDB BUF CONTENTIONS R P
 NO.   OR PSB     TIME     F  CODE    TERMINAL  ID COUNT IN-Q PROC OUTQ TOTAL (SEC) CALL  RD UPD  RD UPD CALL USE CI UW OB BW T T
_____ ________ ____________ _ ________ ________ ___ _____ ____ ____ ____ _____ _____ ____ ___ ___ ___ ___ ____ ___ __ __ __ __ _ _
    1 DDLTJN21 16:02:33.52                    1                                    1   0   0   1   1    0   1  0  0  0  0 B
      BUFFER - NBA=   1 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
    2 DDLTJN21 16:02:33.56                    1                                    6   0   0   2   2    0   2  0  0  0  0 B
      BUFFER - NBA=   2 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
    3 DDLTJN21 16:02:33.57 K                  1                                    5   0   0   1   0    0   3  0  0  0  0
      BUFFER - NBA=   2 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
    4 DDLTJN21 16:02:33.58                    1                                    1   0   0   1   1    0   1  0  0  0  0 B
      BUFFER - NBA=   1 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
    5 DBF#FPUO 16:02:43.55                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
    6 DBF#FPUO 16:02:43.55                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
    7 DBF#FPUO 16:02:43.55                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
    8 DBF#FPUO 16:02:43.55                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
    9 DBF#FPUO 16:02:43.55                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
   10 DBF#FPUO 16:02:43.65                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
   11 DBF#FPUO 16:02:43.65                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
   12 DBF#FPUO 16:02:43.65                    1                                    0   0   0  15   0    0   0  0  0  0  0 U R
      BUFFER - NBA=   0 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   0 NRDB=   0 PBUF=  30 PBWT=   0 ASIO=   0 AIOW=   0
```

Figure 5. DBFULTA: Detail Listing of Exception Transactions—with BUFFER Option Selected

- The region type (BMP, IFP, MPP, or Utility) and the process type are indicated when applicable (HSSP or Reorganization Utility).

The optional print lines, requested by input control statements, are:

1. CALLS

   This line contains counts of all of the different DL/1 calls for the transaction.

2. BUFFER

   This line provides a detailed picture of the transaction′s use of buffers.

   It includes the number of common buffers acquired, within both NBA and OBA. Of these, the number used to provide work areas for SDEP ISRT and MSDB FLD calls is also reported. Other values are the number of times buffer steal was invoked, the number of waits because no buffer was available, and the number of buffers updated and passed to OTHREADs.

   The buffer line also shows buffer usage relevant to HSSP processing, namely, the number of private buffers used and the number of waits for a private buffer.

   Finally, for HSSP Asynchronous Image Copy, the buffer line reports the number of Image Copy read I/Os and the number of HSSP waits for Image Copy to catch up.

```
 SEQ  TRANCODE SYNC POINT  S ROUTING  LOGICAL  PST QUEUE TRANSIT TIMES(MSEC)- -OUT- DEDB ..ADS.. ..VSO.. MSDB BUF CONTENTIONS R P
 NO.   OR PSB     TIME     F  CODE    TERMINAL  ID COUNT IN-Q PROC OUTQ TOTAL (SEC) CALL  RD UPD  RD UPD CALL USE CI UW OB BW T T
_____ ________ __________ _ ________ ________ ___ _____ ____ ____ ____ _____ _____ ____ ___ ___ ___ ___ ____ ___ __ __ __ __ __ _ _
    9 TRANOA01  3:55:38.00   RTCDOA01 ARC05505  52    34   68   20   45   133   0.1    5   1   1   3   3    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   0  REPL   1  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    5
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   1 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    1
   12 TRANOA03  3:55:38.03   RTCDOA03 ARC04387   8    21   73   38   41   152   0.1    9   3   1   4   2    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   3  REPL   2  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    9
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   2 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    0
    7 TRANOA22  3:55:38.12   RTCDOA22 ARC05756  19    14   66   38   48   152   0.1    5   1   1   3   3    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   0  REPL   1  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    5
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   1 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    1
   10 TRANOA22  3:55:38.19   RTCDOA22 ARC05634  42    13   70   23   36   129   0.1    9   3   1   4   2    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   3  REPL   2  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    9
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   2 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    0
    8 TRANOA20  3:55:38.20   RTCDOA20 ARC05975  13     9   72   29   43   144   0.1    5   1   1   3   3    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   0  REPL   1  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    5
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   1 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    1
   14 TRANOA05  3:55:38.24   RTCDOA05 ARC05677  17     6   67   50   45   162   0.1    9   3   1   4   2    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   3  REPL   2  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    9
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   2 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    0
   13 TRANOA22  3:55:38.26   RTCDOA22 ARC05531  21    16   77   30   35   142   0.1    5   1   1   3   3    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   0  REPL   1  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    5
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   1 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    1
   11 TRANOA03  3:55:38.29   RTCDOA03 ARC05660   9    20   79   40   37   156   0.1    9   3   1   4   2    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   3  REPL   2  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    9
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   2 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    0
   15 TRANOA22  3:55:38.32   RTCDOA22 ARC05444  10    15   43   36   24   102   0.1    5   1   1   3   3    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   0  REPL   1  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    5
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   1 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    1
   16 TRANOA03  3:55:38.33   RTCDOA03 ARC05980  34    19   64   33   42   139   0.1    9   3   1   4   2    0   5  0  0  0  0 I
      CALLS  - GU   0  GN   0  GNP   0  GHU   1  GHN   0  GHNP   3  REPL   2  ISRT   1  DLET   0  FLD   2  POS   0  TOTAL    9
      BUFFER - NBA=   5 OVFN=   0 STEAL=   0 WAIT=  0  OTHR=   2 NRDB=   1 PBUF=   0 PBWT=   0 ASIO=   0 AIOW=   0
      VSO    - VGET    3 VPUT    3 DGET    0
```

Figure 6. DBFULTA: Detail Listing of Exception Transactions—with All Three Options Specified

3. VSO

   This line reports for each transaction the number of CIs read directly from a data space, the number of CIs written to a data space, and the number of CIs that had to be read from DASD and copied into a data space.

As in previous releases, there is a code in the basic print line, under the heading "SF," to indicate when syncpoint has failed, with the following possible values:

- **I** For a FLD call at syncpoint time, the required CI was locked by another IMS system (in a data sharing environment) that was no longer available (a retained lock).
- **J** FLD call caused a deadlock at syncpoint time.
- **K** A FLD call failed at syncpoint time because the requested changes failed with arithmetic overflow.
- **M** Reverify failed because the data does not meet verify criteria.
- **N** Reverify failed because the segment moved (for example, because of an REPL call)

Summary of Exception Detail by Transaction Code for IFP Regions: This report has been simplified to include only information that is specific to EMH transactions, namely, the average and maximum values of transit time components and message lengths (see Figure 7 on page 47).

		TRANSIT TIMES IN MILLI-SECONDS								INPUT	MSG	OUTPUT	MSG
TRANS	-NO.OF-	----TOTAL---		--INPUT Q --		--PROCESS --		--OUTPUT Q--		LENG	(CH)	LENG	(CH)-
CODE	-TRANS-	-AVG-	-MAX-	-AVG-	-MAX-	-AVG-	-MAX-	-AVG-	-MAX-	-AVG	-MAX	-AVG	-MAX
TRANOA01	142646	121	219	54	89	24	39	45	63	214	214	488	488
TRANOA03	236589	113	176	42	88	32	44	43	57	184	184	240	240
TRANOA05	48576	145	304	50	105	25	30	39	60	316	316	402	402
TRANOA20	184658	98	158	44	90	19	25	45	61	228	228	596	596
TRANOA22	384620	105	177	41	122	24	36	47	58	196	196	242	242

Figure 7. DBFULTA: Summary of Exception Detail by Transaction Code for IFP Regions

Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs: This report summarizes program resource usage for all application programs (IFP, MPP, DBCTL transactions, BMP, and Fast Path Utilities). Figure 8 shows a sample report.

TRANCODE	--NO.--	------DEDB CALLS-------						-MSDB--		----ADS I/O----				----VSO ACT----				-COMMON BUFFER-			TOTL	CONTENTIONS		TRAN	LGNR	STATS
--OR----	---OF--	-TOTAL-		--GET--		--UPD--		-CALLS-		--RDS--		--UPD--		--RDS -		--UPD--		-----USAGE-----			SYNC	TOT	TOT	CI/	RATE	-NO. OF CI
--PSB---	-TRANS-	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX	WTS	STL	FAIL	UOW	OBA	SEC	/SEC	COMB	LOG'D
DDLTJN21	40	6	30	3	17	0	3	0	0	0	3	0	1	2	27	0	2	4	29	0	0	3	0	0	N/A	N/A	0	0
DBF#FPU0	124	0	0	0	0	0	0	0	0	0	1	0	0	9	15	0	1	0	0	0	0	0	0	0	N/A	N/A	0	0

Figure 8. DBFULTA: Overall Summary of Resource Usage and Contentions for All Transaction Codes and PSBs

Summary of VSO Activity: This is a new report that analyzes VSO activity. It shows, for each VSO area, the number of CIs read from or written to the data space, the number of CIs read from DASD, the number of CIs written back to DASD, and the number of chained I/Os required to do these writes. Figure 9 shows a sample report.

AREA	VSO GETS	VSO PUTS	DASD GETS	DASD PUTS	I/O SCHED
ACDBAR1	2157	1015	0	401	18
ACDBAR2	3255	982	0	951	18
ACDBAR3	2158	2013	14	1014	18
ACDBAR4	1034	31	0	31	2

Figure 9. DBFULTA: Summary of VSO Activity

The main advantages of the revised reports are:

- Detailed information not reported in previous releases is available when needed, as an option. This is particularly important for buffer usage statistics.
- Response time components are more precisely reported.
- The benefits of VSO can be clearly measured.

3.2.15 IOVF Usage Changes

IMS 5.1 improves the use of the independent overflow area for DEDBs. This change improves the throughput capability of an IMS system using DEDBs by reducing contention in a highly parallel environment.

Allocating IOVF CIs

When an IOVF CI is needed to logically extend a UOW, a Space Map[4] is randomly chosen. This algorithm enables parallel allocation of IOVF CIs to requesting programs. Should the chosen Space Map have no available CIs to allocate, a sequential search of all Space Maps is undertaken. Before IMS 5.1, this search started at the first Space Map and hence introduced the possibility of contention if the IOVF were to become full. Because of the potentially greater parallelism in an n-way data sharing environment, the search process has been enhanced by always starting at the Space Map following the Space Map randomized to and wrapping around after the last Space Map in the IOVF.

POS Call and IOVF Statistics

The POS call, in addition to returning the address of an SDEP, returns statistics on the number of unused CIs in the IOVF and SDEP parts of the area referenced. These statistics have previously been kept in memory. However, in an n-way data sharing environment, maintaining the IOVF statistics in every IMS system would be complex and the effort to write such support would be disproportionate to the benefit achieved. (By definition you cannot share DEDBs with SDEPs in IMS 5.1.)

For compatibility reasons, the POS call in IMS 5.1 returns exactly the same values as in previous releases. However, in IMS 5.1 the IOVF count of unused CIs is actually measured at each POS call by reading all of the Space Maps.

Consequently, a POS call will do more I/Os in IMS 5.1 than in previous releases. An additional buffer will be allocated from the region's NBA for this purpose.

The same problem is encountered with a /DIS AREA command, as described in 7.4, "Change to /DISPLAY AREA Command" on page 142.

3.3 MVS Work Load Manager Support

Before MVS 5.1, the way in which work priorities were defined to MVS was extremely complicated and used language far removed from that appropriate for specifying business objectives and Service Level Agreements.

In MVS 5.1 a new work manager, the MVS Work Load Manager (WLM) is available. Business goals are defined to the WLM. MVS endeavors to ensure that the business goals are met by dynamically adjusting resource availability to the many concurrent work processes.

IMS 5.1 interacts with the WLM to make MVS aware of how transactions are currently performing so that system adjustments can be made by MVS to enable the specified transaction goals to be achieved.

The MVS operating system has a large set of functions. These functions create usability problems because of the extraordinary number of parameters that need to be specified, monitored, and tuned to effect the workload mix and prioritization required to meet business objectives.

End-user departments typically enter into Service Level Agreements with the supplier of the computing service, and these agreements specify such things as:

[4] IOVF is in sets of 120 CIs. Each set begins with a Space Map that contains control information about the subsequent 119 CIs.

- Amount of computing resource to be utilized (for example, CPU, storage, DASD space)
- Response time or turnaround times for transactions and jobs
- Availability of the service
- Cost of providing the service.

MVS used to require that business objectives be translated into extremely technical terms in order for MVS to execute them. To specify the service objectives, MVS supplied hundreds of initialization and tuning parameters. The MVS system programmer had to allocate and adjust the amount of computer resources assigned to each user, so that all Service Level Agreements could be reached by manipulating the initialization and tuning parameters. This process can be protracted and error prone and may result in conflict with the original business objectives.

The new MVS WLM introduces a *goal-oriented* management philosophy, allowing the user to explicitly state the system performance expectations, using business language like that used for the Service Level Agreements.

3.3.1 Defining the Business Objectives

MVS supplies an ISPF application (panel driven) that allows a **service definition** to be built. This service definition has a hierarchical structure, as shown in Figure 10 on page 50. Each MVS Sysplex would have its own service definition.

For each service definition there would typically be several service policies. A **service policy** might be created for each distinct time period. For example, separate policies might be defined for the online day, the overnight batch, and the weekend.

Each service policy includes a number of work loads. A **work load** is a business related subset of the applications, for example, an individual application, an application subsystem, or the applications owned by a particular line of business. A typical work load will probably comprise a mixture of transactions (with different degrees of importance), online and offline batch jobs, and house keeping jobs.

Each of these different types of work, with their different values, has a **service class** defined. In other words, a service class equates to a particular type of work with a particular degree of importance to the business. This importance is indicated with a value of 1 (highest importance) to 5 (lowest importance). When the MVS system is overloaded, it decides what to run on the basis of these values.

The IMS system address spaces (control region, DL/1 SAS, and DBRC) will also be assigned a service class. The corresponding performance goal will not be response time related but will relate to how quickly these address spaces will be dispatched when ready to run.

Separately within each service policy are defined a number of performance goals. A **performance goal** can be expressed in several different ways:

1. Response time goal

   For batch work, this is the job elapsed time.

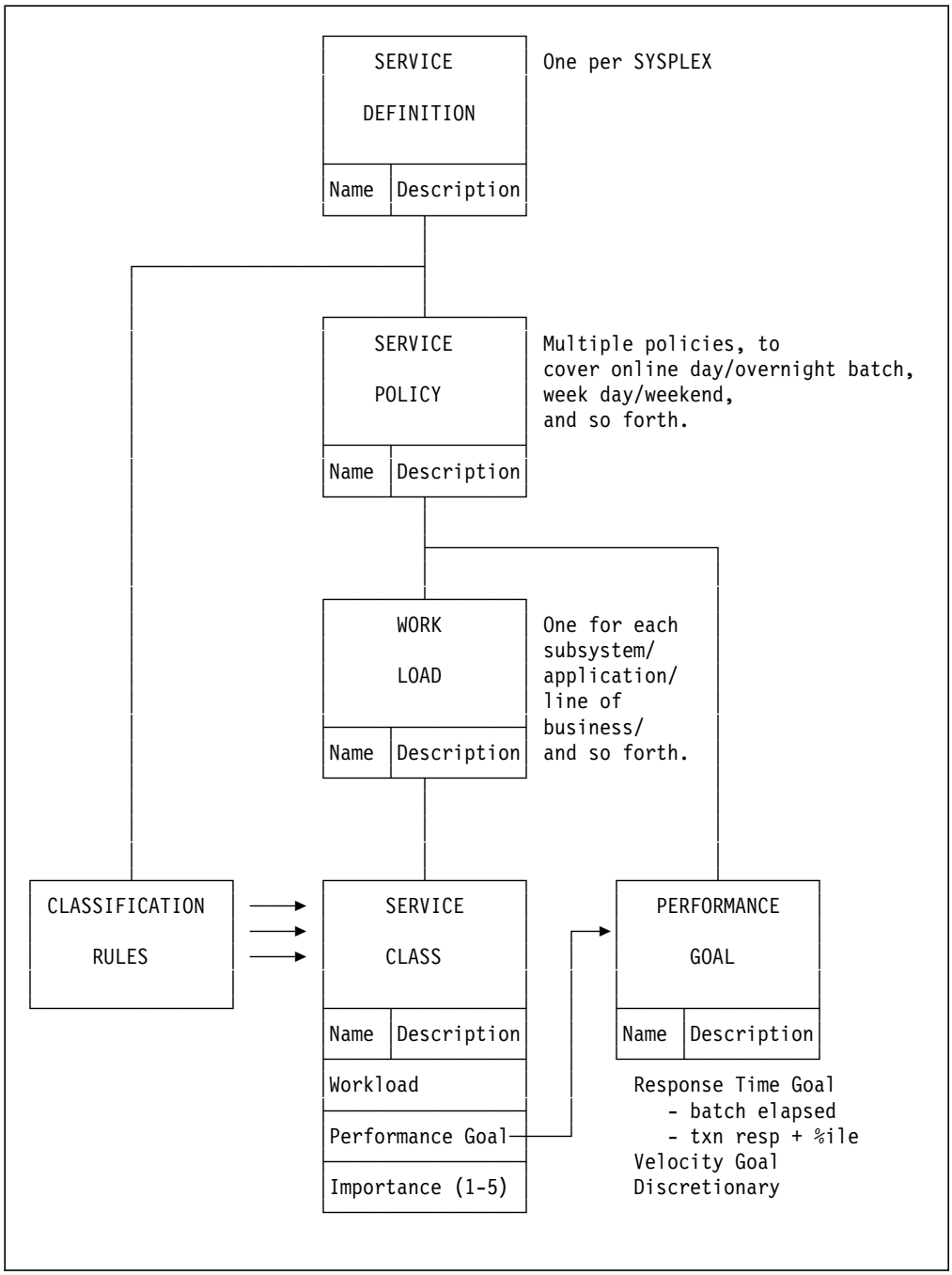


Figure 10. MVS Work Load Manager Definition Hierarchy

For transactions, it is the average response time or the response time specified with a percentile. For example, 95% of transactions are to complete in less than 1.5 seconds.

2. Velocity goal

   This specifies a measure of how fast work should be run when it is ready (dispatchable). For example, you might specify that the IMS Control Region should be immediately dispatched in 90% of cases where it has work waiting to execute.

3. Discretionary goal

This is a way of defining work that can run to “soak up” any spare processing cycles. It will not be run if sufficient system resources are not available. It is unlikely to be used for an online IMS system!

Each service class is associated with one performance goal. It is possible that several service classes will all have the same performance goal, because each work load has its own service classes, but multiple work loads may have high priority transactions, for example.

What remains to be defined is how a service class is assigned to a piece of work. This is accomplished by defining a set of **classification rules**, which define how the various work parameters are used to associate the work with a service class. For example, on a batch job, the jobname, or the userid, or specified accounting information would be appropriate. For an IMS transaction, the service classification can be determined by any combination of:

- IMS ID
- Transaction code
- Transaction class (IMS tells the WLM the current class for a transaction)
- Userid
- LU name
  - In the case of LU6.2, this is the LU name.
  - For ETO, it is the USERNAME (name of USER Structure), which might be the userid, LU name, or any name set by the ETO processing.
  - Otherwise it is the LTERM name.

3.3.2 IMS Interaction with Work Load Manager

When an online IMS system (TM or DBCTL) starts up, it issues a **connect** request to the WLM. At connect time, IMS notifies the WLM of the existence of the three IMS service address spaces (Control, DL/1 SAS, and DBRC). The WLM does not provide an execution monitoring environment for these address spaces; they are treated as service providers to the application programs, which are monitored as explained below. Nevertheless, it is essential that the WLM understands that the IMS address spaces can affect application program performance.

For every local IMS transaction (that is, one that will execute on this system and not on a remote MSC system), IMS calls the WLM to **obtain a service classification**. A service class will be assigned to that individual transaction.

For a non-message-driven BMP or an offline batch job, either MVS or the JES scheduler will obtain the service class.

For DBCTL, the service class is assigned to the CCTL thread associated with the DRA. CCTL (that is, CICS) calls the WLM to get the service class.

IMS Transaction Manager System

IMS calls on the WLM service classification services whenever a message arrives for processing that is:

- A transaction (MPP, IFP, or MD-BMP)
- For this system (that is, not if destination is a remote MSC system)
- From the network or generated by any application ISRT to an alternate PCB.

The WLM monitors the performance of the transaction but relies on IMS to pass it the appropriate information. So, each time IMS issues a send of the

transaction reply (or at syncpoint if no reply is created), it **reports** to the WLM that the transaction has completed. It tells the WLM:

- Time the input message arrived
- Service class that was assigned to the transaction.

The WLM uses current time to calculate the response time.

However, if response performance goals are not being met, the WLM needs to know where delays occur. The WLM is interested in delays due to input queueing, execution time waits (for example, I/O, MVS dispatching, real storage), and output queueing. This information is used by the MVS WLM algorithms to determine where and how to allocate certain MVS controlled resources, such as CPU time and processor storage, to meet the associated performance goals.

Thus IMS must provide information to the WLM about delays. This results in significant numbers of events per transaction having to be reported. To avoid this becoming a measurable overhead, IMS does not directly call the WLM on these occasions. It simply updates a control block, called the *performance block* (PB), associated with the dependent region. Periodically, the MVS WLM looks at the PBs and makes its decisions based on this sampling.

The MVS WLM supplies a set of services called *Delay Monitoring Services* to create, update, and delete the PBs.

A PB is **created** at dependent region startup time and is **deleted** when the region terminates. When a transaction is scheduled, the PB is **initialized** (that is, reinitialized) and will contain a service class token to allow the WLM to associate it with the overall transaction. To update the PB, IMS uses the **change state** service. The PB then reflects the *current state* of the dependent region:

- Free (no transaction currently scheduled)
- Active (transaction executing)
- Idle (transaction scheduled but waiting for work, for example, pseudo-WFI)
- Waiting. The waits reported are:
  - For a lock
  - DB I/O
  - Spool API I/O in the dependent region

At syncpoint time, IMS uses the **notify** service to call the WLM. It passes the PB and indicates that the execution phase of the transaction has finished.

Finally, when IMS is closed down, it issues a **disconnect** request to the WLM.

Table 4 on page 53 summarizes the interactions between the IMS Transaction Manager and the WLM.

Some exceptions to the interactions in Table 4 on page 53 are:

- IMS does not use any WLM services for transactions destined for remote processing through MSC.
- For transactions that do not send a reply to the originating terminal and
  - Do not do a program-to-program switch, IMS reports response time information to the WLM at syncpoint time.
  - Do a program-to-program switch to a transaction with a different service class, IMS reports response time information at syncpoint time. Each transaction in the chain is treated as a new UOW.

Table 4. IMS 5.1 Interactions with MVS Work Load Manager

Event	WLM or PB	Service	Comment
Before transaction processing begins			
IMS startup	WLM	Connect	IMS establishes contact with WLM
Dependent region startup	PB	Create	Builds PB for use by dependent region
During transaction processing			
Message arrival	WLM	Obtain service class	Supplied as a token that is also saved in the PB
Transaction schedule	PB	Initialize	Reinitializes PB for new transaction
Application execution	PB	Change state	Updates PB to indicate active or wait state
Syncpoint	WLM	Notify	Informs WLM that execution phase has finished
Reply sent to terminal	WLM	Report	Informs WLM of the transaction response time and service class
After transaction processing has stopped			
Dependent region termination	PB	Delete	Deletes the PB
IMS close-down	WLM	Disconnect	

Note: The second column (WLM or PB) indicates whether IMS calls the WLM or simply updates the PB

- Do a program-to-program switch to a transaction *with the same service class*, no response time information is reported to the WLM. In this case, the whole chain of transactions will be monitored as a single UOW.

IMS Batch Processing (Including BMPs)

In principle, this is very much like the IMS TM case processing a single transaction. No connect or disconnect is required and the service class is obtained by MVS when the job starts.

MVS creates a PB, but when control is passed to IMS code, it creates a second PB. The MVS PB is the parent PB, and the WLM provides some additional execution monitoring services to enable the parent and dependent PBs to be linked together. IMS uses the **relate** service to make the association apparent to the WLM, and it uses the **transfer** service to notify the WLM that it has started or finished updating the dependent PB (at beginning and end of job, respectively). As the program executes, the dependent PB will be updated by IMS using the change state service. IMS does not need to use the notify service for batch, as it is clear when execution ends.

CICS DBCTL

The DBCTL control address space issues the connect and disconnect just as for the IMS TM case.

CICS obtains the service class, which is assigned to the DRA thread.

IMS creates a PB for the DRA thread and deletes it at thread termination. As in the batch case, this IMS PB is a dependent of a CICS created PB. Consequently, IMS uses the relate and transfer services rather than the initialize and notify services and will use the change state services, as normal, to update the dependent PB.

3.3.3 Implementing IMS 5.1 with the MVS Work Load Manager

In MVS 5.1, both the old (compatibility, IPS, or non-WLM) and the new (WLM) modes of operation are available. The MODIFY command enables a switch to be made from one mode to the other at any time. IMS 5.1 is not dependent on which mode is used. The Connect will always be issued by IMS and always accepted by MVS.

If a service policy is defined, then regardless of the MVS mode, IMS requests and obtains a service class for each transaction and builds and updates a PB in each dependent region. This processing is necessary in case the MVS mode is changed after IMS has been started. IMS will be unaware of the change of mode.

Before the WLM can be exploited for an IMS system, the classification rules must be defined for the IMS transactions. These rules are covered briefly on page 51. As a simplified example, consider a customer who requires the following:

- IMS batch jobs have class J and are to use a different service class from other batch (using the default batch service class)
- All IMS transactions are to have the default service class except for those on IMS4. Of these, transaction codes starting with the characters HIPRI are to be given a different class. (They should also be given a unique IMS scheduling class and dedicated regions. See “WLM and IMS Class Scheduling” on page 55 for an explanation.) Also, all transactions from LTERMs beginning with MGR1 are to be given a special service class.

The classification rules for this scenario are shown in Figure 11 on page 55.

```
Subsystem Type . . . . . JES
Description  . . . . . All batch rules

————Qualifier————    ————Service Class————

                      DEFAULTS:  NORMBTCH
  1   CL      J                  PIMSBTCH


Subsystem Type . . . . . IMS
Description  . . . . . IMS transaction rules


————Qualifier————    ————Service Class————

                      DEFAULTS:  PIMSTRAN
  1   SI      IMS4               IMS4TRAN
   2   TN      HIPRI*            I4HITRAN
   2   LU      MGR1*             I4TOPTRN
```

Figure 11. Example of Specifying IMS Service Classes

An important point to remember is that, when programs are of the same service class and are chained together with multiple program-to-program switches, the goal response time is for the complete set and not for each individual transaction.

Also bear in mind that BMPs use locks and latches and so, just as in the past, should not be defined to the WLM with too low an importance.

WLM and IMS Class Scheduling

The MVS WLM *does not replace* the functions of IMS class and priority scheduling. IMS still selects programs for scheduling. MVS WLM manages the performance of the programs once they are executing (at an address space level).

It is important to recognize that MVS only samples the PBs periodically (a few times per second) and so does not get a detailed view of every individual transaction. For the sampling to provide useful information it is important for IMS to only schedule work of similar profile into a region. Then over time, MVS will get a consistent view of the transactions running in a region and can adjust resource allocation accordingly.

In Figure 11, IMS4 transactions with transaction codes starting with HIPRI are assigned a special service class. For this to work as desired, they would need to be run in dedicated regions so that MVS can monitor their performance without confusion. This would be arranged in the traditional way by allocating a unique class to all of the HIPRI* transactions and setting up separate dependent regions with this same class.

The example in Figure 11 also shows the possibility of defining service classes by LTERM (could also be by USERID). There is no simple way to force IMS to schedule transactions into regions according to the LTERM, and so it is unlikely

that any benefit would be gained by defining a different MVS service class for those LTERMs.

The region class setup and the transaction scheduling data (for example, class, priority, processing limit count) should thus be reviewed when moving to the WLM. Especially in a resource constrained system, it is important to have the IMS class structure set up in a way that matches the relative importance of the transactions as defined in the WLM classification rules. It would be unfortunate if the WLM were withholding resources from what it saw as a low priority transaction, when in fact a high priority transaction was waiting on the queue for that dependent region.

Recommendation

The simplest way to ensure that IMS scheduling and the WLM service classifications are fully compatible is to *define the service classes based on the IMS transaction classes* (that is, not by trancode, LTERM, USERID). For best results, each dependent region should only have classes assigned to it that share the same WLM service classification.

3.3.4 RMF Monitoring with the Work Load Manager

As well as allowing service classes to be defined for different jobs and transactions, you can also define **reporting classes**. In general the reporting classes are aligned with the service classes. The reason for having separately defined reporting classes is to allow, when appropriate, the monitoring of specific terminals or transactions (or a combination of the two) within a service class. In an IMS transaction processing system, given the periodic sampling technique that MVS uses, the response time breakdown information is likely to be suspect for reporting classes that are not aligned with service classes.

RMF reports will show how well (or otherwise) the goals have been achieved. In addition to all of the traditionally useful information on CPU usage, paging, and the like, RMF reports include:

- Velocity goal and achievement for IMS system address spaces (Control region, DL/1 SAS, and DBRC)
- A list of transaction service classes being serviced by that IMS. For each of these service classes, it reports:
  - Transaction rate
  - Response time goals and achievements
  - Response time distribution
  - Breakdown into queueing time and execution time
  - Breakdown of execution time delays.

Figure 12 on page 57 shows a sample RMF Workload Activity Report for IMS system address spaces, and Figure 13 on page 58, for IMS dependent regions. This is not meant to be a tutorial in interpreting RMF reports, but a few items are worth highlighting in the two reports.

In Figure 13 on page 58, the IMS system address spaces had a velocity goal of 80% but an achievement of 92.3%. The service classes being served are just one in number, namely IMSTRX.

In Figure 13 on page 58, 10,826 transactions ran at an average rate of 12.03 per second. The transaction goal was an average response time of 0.1 seconds, but the achieved result was 0.136 seconds, as reported to the WLM by the IMS control region. Of this, 0.108 seconds was execution time and 0.028 seconds was "queueing time." So 79.4% of the response time was spent in program execution (108 msec is 79.4% of 136 msec).

Referring to the section of the report entitled "RESPONSE TIME BREAKDOWN IN PERCENTAGE," the total percentage of response time (136 msec) for which sampling has been performed is 72.7%. The difference between this and the 79.4% discussed above should not cause concern and is due to the sampling technique, which can only recognize a PB that is initialized *after* it has become initialized.

Of the 72.7% of response time that was sampled, the program was executing 70.9% of the time, waiting for a lock 1.7% of the time, and I/Os only accounted for 0.1% of the time. (One could perhaps conjecture that this system was using the new DEDB VSO option!)

```
                                    W O R K L O A D   A C T I V I T Y
                                                                                                        PAGE   8
      MVS/ESA                SYSPLEX UTCPLXJC          DATE 03/29/1994          INTERVAL 14.59.579   MODE = GOAL
      SP5.1.0                RPT VERSION 5.1.0         TIME 14.30.11

                                POLICY ACTIVATION DATE/TIME 03/28/1994 16.06.26

------------------------------------------------------------------------------------------------------------------ SERV. CLASS PERIOD(S)

REPORT BY: POLICY=WSTPOL01   WORKLOAD=IMS        SERVICE CLASS=IMSREGS    RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=HIGH

-TRANSACTIONS--  TRANSACTION TIME   HHH.MM.SS.TTT   INTERVAL SERVICE   --SERVICE RATES--   -PAGE-IN RATES--   ----STORAGE----
AVG       47.00  ACTUAL             000.00.00.000   IOC      432989    ABSRPTN      346    SINGLE      0.00   AVG      490.46
MPL       47.00  QUEUED             000.00.00.000   CPU     4734703    TRX SERV     346    BLOCK       0.00   TOTAL   23052.9
ENDED         0  EXECUTION          000.00.00.000   MSO     8649823    TCB        913.9    HSP         0.00   CENTRAL 19996.5
END/SEC    0.00  STANDARD DEVIATION 000.00.00.000   SRB      826021    SRB        159.4    HSP MISS    0.00   EXPAND  3056.44
#SWAPS        0                                     TOT       14644K   TCB+SRB%     119    EXP SNGL    0.00
EXECUTD       0                                     PER SEC   16278                        EXP BLK     0.00

        ---RESPONSE TIME---  EX   PERF   AVG   USING  ----------------- EXECUTION DELAYS % ------------------ ---DELAY  %---
        HH.MM.SS.TTT         VEL  INDX  ADRSP  CPU %  TOTAL  CPU CAPP SWIN  MPL  -AUXILIARY PAGE DELAY % FROM-  OTHR IDLE QUIE
GOALS                       80.0%                                              PRIV COMM XMEM VIO  SHSP EHSP
ACTUALS                     92.3%  0.9   47.0   3.0     0.3  0.3  0.0  0.0  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  93.9  2.8  0.0

------------------------------------------------------------------------------------------------------------------ SERVICE CLASS(ES)

REPORT BY: POLICY=WSTPOL01   WORKLOAD=IMS        SERVICE CLASS=IMSREGS    RESOURCE GROUP=*NONE
                                                          Solution Test - IMS Regions

-TRANSACTIONS--  TRANSACTION TIME   HHH.MM.SS.TTT   INTERVAL SERVICE   --SERVICE RATES--   -PAGE-IN RATES--   ----STORAGE----
AVG       47.00  ACTUAL             000.00.00.000   IOC      432989    ABSRPTN      346    SINGLE      0.00   AVG      490.46
MPL       47.00  QUEUED             000.00.00.000   CPU     4734703    TRX SERV     346    BLOCK       0.00   TOTAL   23052.9
ENDED         0  EXECUTION          000.00.00.000   MSO     8649823    TCB        913.9    HSP         0.00   CENTRAL 19996.5
END/SEC    0.00  STANDARD DEVIATION 000.00.00.000   SRB      826021    SRB        159.4    HSP MISS    0.00   EXPAND  3056.44
#SWAPS        0                                     TOT       14644K   TCB+SRB%     119    EXP SNGL    0.00
EXECUTD       0                                     PER SEC   16278                        EXP BLK     0.00

-------------------------------------------------SERVICE CLASSES BEING SERVED-------------------------------------------------
IMSTRX
```

Figure 12. RMF Workload Activity Report for IMS System Address Spaces: Control Region, DL/1 SAS, and DBRC

```
                                    W O R K L O A D   A C T I V I T Y
                                                                                                                PAGE   9
      MVS/ESA                 SYSPLEX UTCPLXJC          DATE 03/29/1994          INTERVAL 14.59.579   MODE = GOAL
      SP5.1.0                 RPT VERSION 5.1.0         TIME 14.30.11

                                       POLICY ACTIVATION DATE/TIME 03/28/1994 16.06.26

------------------------------------------------------------------------------------------------------------------- SERV. CLASS PERIOD(S)

REPORT BY: POLICY=WSTPOL01   WORKLOAD=IMS        SERVICE CLASS=IMSTRX     RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=HIGH

-TRANSACTIONS--  TRANSACTION TIME   HHH.MM.SS.TTT
AVG        0.00  ACTUAL             000.00.00.136
MPL        0.00  QUEUED             000.00.00.028
ENDED     10826  EXECUTION          000.00.00.108
END/SEC   12.03  STANDARD DEVIATION 000.00.00.160
#SWAPS        0
EXECUTD   10826

           -------------------------------RESPONSE TIME BREAKDOWN IN PERCENTAGE----------------------------   ------STATE------
SUB    P   TOTAL  ACTIVE  READY   IDLE   ----------------------------WAITING FOR----------------------------   SWITCHED TIME (%)
TYPE                                     LOCK    I/O   CONV   DIST  LOCAL  SYSPL  REMOT  TIMER   PROD   MISC   LOCAL SYSPL REMOT
IMS   EXE   72.7    70.9    0.0    0.0    1.7    0.1    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0     0.0   0.0   0.0

        ---RESPONSE TIME---  EX   PERF
        HH.MM.SS.TTT         VEL  INDX
GOALS   00.00.00.100   AVG
ACTUALS 00.00.00.136         0.0%  1.4

-----------------------------------------------------RESPONSE TIME DISTRIBUTION-----------------------------------------------------
   ----TIME----    ---NUMBER TRANSACTIONS---           ----PERCENT----  0    10   20   30   40   50   60   70   80   90   100
   HH.MM.SS.TTT    BUCKETS             TOTAL          BUCKETS   TOTAL  |....|....|....|....|....|....|....|....|....|....|
<  00.00.00.050       3717              3717             34.3    34.3  >>>>>>>>>>>>>>>>>
<= 00.00.00.100       2254              5971             20.8    55.2  >>>>>>>>>>>>>>>>>>>>>>>>>>>>
<= 00.00.00.200       2540              8511             23.5    78.6  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>  00.00.00.200       2315             10826             21.4     100  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

------------------------------------------------------------------------------------------------------------------- SERVICE CLASS(ES)

REPORT BY: POLICY=WSTPOL01   WORKLOAD=IMS        SERVICE CLASS=IMSTRX     RESOURCE GROUP=*NONE
                                                                  IMS Transactions

-TRANSACTIONS--  TRANSACTION TIME   HHH.MM.SS.TTT
AVG        0.00  ACTUAL             000.00.00.136
MPL        0.00  QUEUED             000.00.00.028
ENDED     10826  EXECUTION          000.00.00.108
END/SEC   12.03  STANDARD DEVIATION 000.00.00.160
#SWAPS        0
EXECUTD   10826
```

Figure 13. RMF Workload Activity Report for IMS Dependent Regions

3.3.5 Value

Clearly, the WLM will be the only way in the future of allocating system resources. Its role will be crucial in managing the work distribution across Parallel Sysplexes. But even on a single MVS system, it can provide significant benefit.

The WLM way of defining the business importance of transactions and batch work is simpler and less error prone than before and consequently more likely to achieve the desired results.

Using RMF, it is very easy to monitor performance against goals.

3.4 DB2 Pseudo-Wait for Input

The DB2 Pseudo-WFI is an optional facility that improves the throughput of an IMS system and reduces the transaction processing cost by reducing the number of IOs and processor time used, while improving the transaction response time. Although the saving is small, it is noticeable and has a negligible effect in most systems.

Pseudo-WFI is a dependent region option with the potential to reduce the number of program fetches and hence transaction processor and elapsed time. Before IMS 5.1 this facility could only benefit transaction programs that did not access DB2.

In IMS 5.1, programs accessing DB2 are treated exactly like non-DB2 programs. They can benefit from the reduced program loading and gain additional processor savings because of the reduced number of DB2 thread creations.

Pseudo-Wait for Input was introduced with IMS 3.1. It is an option that is requested as a dependent region JCL EXEC parameter (PWFI=Y|N). With this option, IMS does not tell the program that there are no more messages immediately after all the transactions in the queue have been processed. Instead, IMS returns to the scheduler to see whether a different program needs to be loaded in that region. If so, the original program is notified that there are no more messages (QC status code to IOPCB GU), it terminates, and IMS loads the new program. However, if there is no other work waiting for that region, IMS does nothing until further transactions arrive in the system. When a transaction arrives, IMS looks to see whether the transaction program is already loaded and in the pseudo-WFI state. If so (and if not restricted by parallel processing constraints), the new message is passed to the waiting application and so processed without the overhead of program fetch. To exploit this facility fully, IMS tends to use *all* available Pseudo-WFI regions, regardless of system load, in order to maximize the number of loaded programs executing or waiting to execute.

However, before IMS 5.1, if a program running in a Pseudo-WFI region accessed DB2, the Pseudo-WFI facility was temporarily disabled. When that program issues its next IOPCB GU and there are no messages, a QC status is immediately returned to the application. The consequence is that more program schedules are required, and more program fetches take place. Program fetch is a significant user of processor and program library I/O (if facilities like LLA/VLF are not used).

To process DB2 calls, a dependent region must connect to DB2, and this is done once for the life of the region. Each time a program is scheduled, a DB2 thread must be created (and terminated at program termination). Each time a new message is accessed by the application program, a DB2 signon must take place to identify the transaction′s userid and group name to DB2.

A consequence of not having Pseudo-WFI for DB2 transactions is that there are more create threads. Thread creation is a relatively heavy processor intensive process (though significantly reduced in IMS 5.1—see 3.5, “DB2 Access from IMS Applications” on page 61), and so again the lack of Pseudo-WFI adds to the processor cost.

In IMS 5.1, the Pseudo-WFI option applies to all transactions, regardless of whether they access DB2 or not. For transactions accessing DB2, Pseudo-WFI offers the potential to reduce the processor cost (and elapsed time) by reducing the number of program loads and create threads.

3.4.1 Implementing Pseudo-WFI with DB2

Other than ensuring that the dependent region JCL specifies PWFI=YES, nothing else is required to implement Pseudo-WFI.

However, some customers may find it possible to specifically exploit Pseudo-WFI, by increasing the number of regions in which the DB2 transactions can run. (Note that IMS 5.1 has increased the maximum number of dependent regions beyond 255. See 7.2, “Up to 999 Dependent Regions or CICS Threads” on page 140.) The feasibility of this depends on many factors including:

- The number of active transaction codes

  A customer with 4000 equally common transaction codes processing in 20 regions is not likely to be able to make significant use of Pseudo-WFI.

- Application real storage requirement

  Each dependent region must be backed up by real storage. The savings from avoiding program load are likely to disappear if the program has to be paged-in.

- PSB pool size

  If many programs are predominantly or exclusively DB2 based, the additional PSB pool requirements will be modest. But if significant numbers of DL/1 databases are also used, the extra PSB pool space could be considerable and must be backed up by real storage.

- From the DB2 perspective, each loaded program requires a plan in the EDM pool, which also has storage implications.

As always, increasing the number of regions should be done in steps to ensure that resources are not overcommitted. At each step the benefits should be monitored and understood. Also, as already mentioned, the parameters relating to parallel scheduling (PARLIM and MAXRGN) should be reviewed.

3.4.2 Value

The main benefit of Pseudo-WFI for DB2 is the potential reduction in processor time per transaction, which translates into reduced cost per transaction. In systems using traditional program fetch from a program library, there also can be a significant reduction in average elapsed time per transaction.

The customer who would see the most benefit would be one using a relatively small number of large applications (long load times) where transactions arrive not quite frequently enough to fully exploit the PROCLIM facility. Most customers are not like this, but will nevertheless usually gain some degree of benefit. And there is nothing to lose by trying Pseudo-WFI.

3.5 DB2 Access from IMS Applications

The improvement in DB2 usage can be great, particularly for short transactions where the improvement can be a significant percentage of the total transaction cost.

3.5.1 Overview

Application access to DB2 requires one Create Thread per program schedule and a DB2 SIGNON for each message processed. In IMS 5.1, these two processes have been made significantly more efficient.

3.5.2 Description

For an IMS application program that accesses DB2, at the first SQL call after program schedule, the dependent region has to create a thread to DB2 and then signon the current user (for example, terminal user) to DB2. For each subsequent transaction that the program processes, the new user has to be identified to DB2 by signon.

SQL calls are issued while the program is not in an MVS authorized state. However, both create thread and DB2 signon require the caller to be running authorized. In previous releases of IMS, this was achieved by issuing a subsystem interface call and an SVC. This results in additional processor usage and a hardware interrupt, which adds to the transaction cost. In IMS 5.1 the change of state from unauthorized to authorized has been rewritten, and the subsystem interface call and SVC have been replaced by a PC (program call) instruction.

During these processes in releases earlier than IMS 5.1, register save areas needed to be acquired dynamically with IMODULE GETMAINs, which used SVCs and again had associated processor cost. IMS 5.1 makes use of the very efficient MVS Cell Pool facility. When the region connects to DB2 at dependent region startup time, a CPOOL macro is issued to build a cell pool of the appropriate size (with a secondary allocation defined in case of exceptional requirements). This pool will be used for the register save areas needed at create thread and signon times.[5]

3.5.3 Value

All transactions that access DB2 will now use less processor time. The saving is unrelated to the number of SQL calls.

[5] The IMS DB2 interface first made use of the Cell Pool facility in IMS 4.1, where, since APAR PN47032, all SQL calls use CPOOL storage for work area space (but not for thread creation or sign-on)

3.6 Enhanced Log Formatting and Select Utility

The improved ease of use in locating units of recovery can lead to a time saving in certain recovery situations, which makes this utility enhancement a small but important improvement in the availability of IMS.

3.6.1 Overview

The File Select and Formatting Print Utility (DFSERA10) has received two enhancements. Both changes apply to the use of the Enhanced Select Exit Routine (DFSERA70):

- Standard *fixed* format for prints of database log records
- *All* log records associated with a specified unit of recovery (UOR) can be printed.

These changes will simplify the analysis of problems or issues relating to database updates or complete UORs.

3.6.2 Description

The File Select and Formatting Print Utility (DFSERA10) is well known to both IMS and CICS/DL1 users for its powerful parameter-driven log record selection and printing facilities. It has proved a most useful tool in expediting resolution of many problems where the solution lies in the IMS log records. The two enhancements provided in IMS 5.1 make the utility even more usable and appropriate problem resolution both easier and quicker.

Several subroutines are provided with IMS to print selected log or trace records in a predefined manner, and these are summarized in Table 5.

Table 5. DFSERA10 Exit Routines

Exit	Function
DFSERA30	Used to print trace, snap, and abend records. A particularly useful feature of the abend log record handling is that a *Deadlock Analysis Report* will be printed where appropriate.
DFSERA40	Formats the program isolation (PI) trace records (for either PI or IRLM)
DFSERA50	Formats DL/1 call trace records for either printing or inputting to the DL/1 test program (DFSDDLT0)
DFSERA60	Used to print all (or any) trace records from the log or trace data sets
DFSERA70	Provides enhancements to the basic DFSERA10 facilities: • For database update log records, references fields by name rather than offset. • Prints log records relating to a specific UOR

In IMS 5.1, the functions of the DFSERA70 exit have been further enhanced.

Database change log records (Types 50 and 5950) can be printed in a new, *standard format*. Both print formats are shown in Figure 14 on page 63.

In the new format, the basic log record data is printed first, followed by the actual database data. The content of the database data depends on the nature of the system and the DL/1 call. Only relevant types of data are printed. The

possible types relate to data sharing, XRF buffer and lock tracing, space management, key (KSDS), backout (undo), and recovery (redo). Examples of each of these (except for XRF) are shown in Figure 15 on page 64.

For reasons of compatibility, this new facility is a user option requested with a new parameter, XFMT=YES. For example:

```
OPTION  PRINT  EXITR=DFSERA70,PARM=(XFMT=YES)
```

```
OPTION  PRINT  E=DFSERA70


50 RECORD
 00000000 000000   008E0000 5050000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C586 39ED9186  *....&&..CICS2   ZTE,.M..ZTEF..JF*
 00000020 000020   8094208F 1435386F  43002040 C8C8E3C1   E2D2F4F4 C4C8F4F1  E2D2F0F1 01038000  *.M.....?... HHTASK44DH41SK01....*
 00000040 000040   00000002 00000000  000000D8 0000005E   00000000 00000000  0000006C 00760000  *...........Q...;...........%....*
 00000060 000060   00010000 00000000  00000001 804000D8   00040000 0D408040  00D80004 00000E58  *.............. .Q..... . .Q......*
 00000080 000080   00000004 146B0000  00000000 0076                                           *.....,........                *


OPTION  PRINT  PARM=(XFMT=Y),E=DFSERA70


50 RECORD
 00000000 000000   005E0000 5050000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C583 A575E582  *.;..&&..CICS2   ZTE,.M..ZTECV.VB*
 00000020 000020   8094208F 1435357F  43002040 C8C8E3C1   E2D2F4F4 C4C8F4F1  E2D2F0F1 01038000  *.M.....”... HHTASK44DH41SK01....*
 00000040 000040   00000002 00000000  0000000A 0000005E   00000000 00000000  0000006C 0076      *...............;...........%..  *
DSHRDSSN  00000001  DSHRLSN   000000000000  DSHRUSID  00000001  RACF-UID
UNDO  PHYSICAL REPLACE                      LENGTH    0004  OFFSET    000A
 0000000A 000000   00000000                                                                  *....                           *
REDO  PHYSICAL REPLACE                      LENGTH    0004  OFFSET    000A
 0000000A 000000   00000DCC                                                                  *....                           *
```

Figure 14. DFSERA70 XFMT Option

The other DFSERA70 enhancement is when using the TOKEN parameter:

```
OPTION PRINT EXITR=DFSERA70,PARM=(TOKEN=x...x)
```

Previous releases of IMS only printed database change records. IMS 5.1 prints *all* log records for the UOR. These include X′07′, X′08′, X′0A′, X′13′, X′27′, X′28′, X′31′, X′32′, X′35′, X′37′, X′38′, X′39′, X′3D′, X′41′, X′4C′, X′50′, X′56′, X′5901′, X′5903′, X′5937′, and X′5938′. If it is required to show only the log records that would have been selected in a previous release of IMS, simply include DFSERA10 selection criteria for type 50 (and/or 5950 for DEDB) records.

```
OPTION PRINT OFFSET=5,VALUE=50,EXITR=DFSERA70,PARM=(TOKEN=x...x)
```

Figure 15 on page 64 is an extract from the output of running DFSERA70 to process a CICS DBCTL log with both TOKEN and XFMT specified.

3.6.3 Implementing DFSERA70 Enhancements

The new formatting of database log records is requested by including the XFMT=YES parameter.

The UOR analysis enhancements require no new or changed parameters.

3.6.4 Value

The DFSERA10 utility with the DFSERA70 exit is typically used to help in resolving problems. Quite often, such activity is done under stress, with pressures to find a solution *immediately*. These two enhancements make the task simpler and more efficient.

```
CONTROL  CNTL  STOPAFT=EOF
OPTION  PRINT  PARM=(XFMT=Y,TOKEN=X'A9A3C56BDFD4BA06'),E=DFSERA70
END


08 RECORD
 00000000 000000   005C0000 0804C8C8  E3C1E2D2 F4F44040   40404040 40400094  208F1435 230F0003  *.*....HHTASK44        .M........*
 00000020 000020   000BC3C9 C3E2F240  4040A9A3 C56BDFD4   BA060000 00020000  00000001 00000000  *..CICS2   ZTE,.M..............*
 00000040 000040   0000000C 00000000  0000000C 00000001   0967289C 00000000  00000060           *.........................-    *

56 RECORD
 00000000 000000   004C0000 56070000  E2E8E2F1 40404040   0000000B C8C8E3C1  E2D2F4F4 00000000  *.<......SYS1    ....HHTASK44....*
 00000020 000020   00000000 0094208F  1433394F C3C9C3E2   F2404040 A9A3C56B  DFD4BA06 00000000  *.....M.....|CICS2   ZTE,.M......*
 00000040 000040   00000000 00000000  00000061                                                *...........|                    *

50 RECORD
 00000000 000000   005E0000 5050000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C583 5B4A7380  *.;..&&..CICS2   ZTE,.M..ZTEC$...*
 00000020 000020   8094208F 1435357F  57002040 C8C8E3C1   E2D2F4F4 C4C8F4F1  E2D2F0F1 01038000  *.M.....".... HHTASK44DH41SK01....*
 00000040 000040   00000002 00000000  44000090 0000005E   00000000 00000000  006C0000 0072      *.............;.........%....  *
DSHRDSSN  00000001  DSHRLSN   000000000000  DSHRUSID  00000001  RACF-UID
SMGTFLGS  20  00    SMGTROFF  0090          SMGTRLEN  0046
REDO  FREE SPACE ELEMENT                    LENGTH    0004  OFFSET    0000
 00000000 000000   00D60000                                                                  *.O..                            *
REDO  FREE SPACE ELEMENT                    LENGTH    0004  OFFSET    00D6
 000000D6 000000   000005C8                                                                  *...H                            *

50 RECORD
 00000000 000000   005E0000 5050000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C583 A572C902  *.;..&&..CICS2   ZTE,.M..ZTECV.I.*
 00000020 000020   8094208F 1435357F  43002040 C8C8E3C1   E2D2F4F4 C4C8F4F1  E2D2F0F1 01038000  *.M.....".... HHTASK44DH41SK01....*
 00000040 000040   00000002 00000000  40000090 0000005E   00000000 00000000  00000000 006C      *........ ......;............%  *
DSHRDSSN  00000001  DSHRLSN   000000000000  DSHRUSID  00000001  RACF-UID
REDO  PHYSICAL INSERT                       LENGTH    0046  OFFSET    0090
 00000090 000000   01000000 0D400000  00000000 00000000   00000000 00000000  00000000 0000C1F0  *..... .......................A0*
 000000B0 000020   C1F0F0F0 F0F1F1F2  F1F1F1F1 F1F1F1F1   F1F1F1F1 F1F1F1F1  F1F1F1F1 F1F1F1F1  *A000011211111111111111111111111111*
 000000D0 000040   F1F1F1F1 F1F1                                                              *111111                          *

50 RECORD
 00000000 000000   005E0000 5050000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C583 A575E582  *.;..&&..CICS2   ZTE,.M..ZTECV.VB*
 00000020 000020   8094208F 1435357F  43002040 C8C8E3C1   E2D2F4F4 C4C8F4F1  E2D2F0F1 01038000  *.M.....".... HHTASK44DH41SK01....*
 00000040 000040   00000002 00000000  0000000A 0000005E   00000000 00000000  0000006C 0076      *.............;..........%..  *
DSHRDSSN  00000001  DSHRLSN   000000000000  DSHRUSID  00000001  RACF-UID
UNDO  PHYSICAL REPLACE                      LENGTH    0004  OFFSET    000A
 0000000A 000000   00000000                                                                  *....                            *
REDO  PHYSICAL REPLACE                      LENGTH    0004  OFFSET    000A
 0000000A 000000   00000DCC                                                                  *....                            *

50 RECORD
 00000000 000000   005E0000 5052000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C586 2D36D180  *.;..&...CICS2   ZTE,.M..ZTEF..J.*
 00000020 000020   8094208F 1435386F  53000884 C8C8E3C1   E2D2F4F4 C4E7F4F1  E2D2F0F1 01038000  *.M.....?...DHHTASK44DX41SK01....*
 00000040 000040   00000000 00000000  42000000 0000005E   00000000 0000006C  0000007A 0000      *.............;.......%...:..  *
DSHRDSSN  00000001  DSHRLSN   000000000000  DSHRUSID  00000001  RACF-UID
KSDS                                        LENGTH    000A
 0000006C 000000   C1F0C1F0 F0F0F0F1  F1F2                                                    *A0A0000112                      *
UNDO  PHYSICAL INSERT                       LENGTH    0014  OFFSET    0000
 00000000 000000   0000000D CCC1F0C1  F0F0F0F0 F1F1F200   00000000                           *.....A0A0000112.....            *

50 RECORD
 00000000 000000   005E0000 5050000B  C3C9C3E2 F2404040   A9A3C56B DFD4BA06  A9A3C586 2D44C500  *.;..&&..CICS2   ZTE,.M..ZTEF..E.*
 00000020 000020   8094208F 1435386F  43000884 C8C8E3C1   E2D2F4F4 C4E7F4F1  E2D2F0F1 01038000  *.M.....?...DHHTASK44DX41SK01....*
 00000040 000040   00000000 00000000  40000000 0000005E   00000000 0000006C  00000000 007A      *........ ......;.......%.....:  *
DSHRDSSN  00000001  DSHRLSN   000000000000  DSHRUSID  00000001  RACF-UID
KSDS                                        LENGTH    000A
 0000006C 000000   C1F0C1F0 F0F0F0F1  F1F2                                                    *A0A0000112                      *
REDO  PHYSICAL INSERT                       LENGTH    0014  OFFSET    0000
 00000000 000000   0000000D CCC1F0C1  F0F0F0F0 F1F1F200   00000000                           *.....A0A0000112.....            *
Using XFMT=YES with a DBCTL Log
```

Figure 15. DFSERA70 with TOKEN Parameter

For example, suppose a data error is discovered on a database. DFSERA10 with the DFSERA70 exit provides a simple way to discover which program made the erroneous update (using parameters such as DBD=dbdname, RBA=cirba, OFFSET=segaddr, REDO=baddata, or, even more powerfully, DATA=baddata). This will provide the recovery token, which can be used on a second run of the utility to format and print all of the log records for that UOR. The printed log records will usually be sufficient to allow the cause of the problem to be identified (though it should be noted that the log records that contain the actual

message text (X′01′ for input, X′03′ for output) are not part of the UOR set of log records).

3.7 Virtual Storage Constraint Relief

Virtual storage constraint relief (VSCR) is given a big boost in IMS 5.1 with a very significant reduction in below-the-line common storage usage. This enhancement is particularly significant for customers executing many IMS images in a single MVS (for example, in an application development environment).

3.7.1 Overview

Some IMS customers (DBCTL and Transaction Manager) are still experiencing virtual storage constraint in the MVS CSA. To provide further relief, IMS 5.1 has moved a number of modules from CSA into Extended CSA (ECSA), reducing the CSA requirement by about 100KB.

3.7.2 Description

For IMS 5.1, the VSCR strategy was to examine the code modules that resided in CSA and identify those that (a) could be modified to reside in ECSA and (b) would provide significant constraint relief. Nearly 30 such modules were identified and updated to reside in ECSA. These modules are from various IMS components, including Fast Path, DBCTL, and XRF. Consequently, the large, complex systems will see the most relief, up to about 128KB. Less complex systems and DBCTL can perhaps expect CSA relief of about 100KB.

3.8 OSAM Dynamic Cache Management Enhanced Support

The Dynamic Cache Management Enhanced (DCME) support for OSAM exploits the capabilities of 3990 control units with cache in a similar way that VSAM provides. It can result in significant reductions in access time for OSAM databases in most cases. This can increase the throughput and peak capacity of the IMS system if the limiting factor is the database access time as well as improve the response time for transactions using these databases.

3.8.1 Overview

DCME is a DFSMS service that provides the potential to gain performance benefits by making better use of cached 3990 DASD Control Units. Before IMS 5.1, only VSAM data sets could exploit DCME. In IMS 5.1, OSAM data sets can also gain the performance benefits.

3.8.2 Description

DCME is provided in DFSMS/MVS and has the objective of relieving the customer storage administrator of cache tuning responsibilities. It works in conjunction with Licensed Internal Code on the 3990 to provide data set level caching. With the newer 3990 models 6 and 9, record caching (as well as track caching) is supported. Initially, this requires DCME. The second phase of record caching, due in early 1995, does not rely on MVS software to enable record level caching but will “take advice” from DFSMS DCME if offered.

By supporting DCME for OSAM in IMS 5.1, both VSAM and OSAM database data sets can make full use of data set level caching, record caching, and dynamic cache management (automatic choice of track, record, or no caching).

3.8.3 Value

Transaction processing generally requires little or no sequential I/O. With track caching, a reference to a block or a CI will cause the rest of the track to be staged into the 3990 cache. In general this will not be referenced before it ages out of the cache. So the track staging is wasteful, can interfere with other I/Os to the same volume, and can degrade the performance of a busy 3990. There will be some cases where the cached data is referenced (read hit), and writes are almost assured of a write hit with consequential DASD Fast Write. Also, BMPs may well do sequential processing and get good read hit rates. So, in many cases, track level caching does provide an overall benefit. But some customers have been forced to turn caching off for certain database volumes, as the benefits were outweighed by the performance costs.

Record caching does not stage in the rest of track after a block has been read into cache and so is very attractive for the random I/Os characteristic of online transaction processing. DASD Fast Write will still be achieved (in fact is guaranteed with record caching) and dynamic cache management will switch to track caching should it become appropriate, as for sequential BMPs.

DCME has supported VSAM database data sets since it became available. IMS 5.1 provides the same benefits to users of OSAM. So the OSAM DCME support has the potential to improve the efficiency of cache usage, which will enable better overall I/O subsystem performance.

Those customers who have been forced to turn caching off for OSAM databases will, with IMS 5.1, be able to use caching and gain the benefits of DASD Fast Write, and perhaps the occasional fast read. Sequential BMPs and batch jobs will automatically utilize track caching, and so they too will benefit significantly.

Chapter 4. IMS 5.1 Operating Cost Enhancements

This chapter describes the new IMS Version 5.1 features that can simplify operations and so reduce the cost of operating an IMS system.

4.1 New Automated Operator Interface

The need for automating IMS operations is well understood, especially when networks become larger and larger and operations becomes ever more complex.

For many years, automated operator functions for IMS were provided by the Automated Operator Exit (DFSAOUE0) and AOI transactions. In IMS 3.1, Time-Controlled Operations (TCO) became available for a DB/DC environment.

IMS 5.1 includes a new facility that provides DBCTL users with a way of automating their IMS system operations. It is designed to work without the message queues. Instead it makes use of an incore queue. For users of the IMS TM, this new automated operator interface provides automation capability even if the message queue is not accessible. IMS TM users can use either the old or the new automation facility.

4.1.1 Overview

The IMS Automated Operator Interface is a set of functions that allows more efficient operation of the IMS system. As mentioned earlier, there are now in IMS 5.1 two AOI facilities:

1. AOI - Type 1, the AOI facility provided in earlier IMS releases, which exists for the IMS TM environment only.

   The AO exit routine used is the Automated Operator Exit - Type 1 (DFSAOUE0), and it uses the IMS message queues for inserting messages to transactions or LTERMs.

   An AO application corresponding to such a transaction can:

   - Issue GU/GN calls to retrieve messages routed by the AO exit through the IMS message queue
   - Issue DL/1 CMD calls to issue operator commands
   - Issue DL/1 GCMD calls to retrieve command responses

2. AOI - Type 2, new in IMS 5.1, which exists for both TM and DBCTL environments

   This new AOI contains:

   - A new user exit routine, Automated Operator Exit - Type 2 (DFSAOE00)
   - New DL/1 calls for use by AO applications.

   For communication between the AO exit and the AO applications, a user-defined AOI token and an in-memory table (instead of the IMS message queues) are used.

The purpose of the exit is to intercept:

- IMS system messages routed to the master terminal for IMS TM or to the MVS system console in the DBCTL environment
- IMS commands
- IMS command responses.

After examination of these intercepted messages, the exit can send a message to a new type-2 AO application but without using the message queues.

The AO Application - Type 2 can then:

- Retrieve messages from the Automated Operator Exit - Type 2 (DFSAOE00)
- Issue various IMS operator commands
- Retrieve responses to these commands.

In both cases, the AO exit and the AO application can be used together or separately, depending on the need of the installation (see Figure 16).

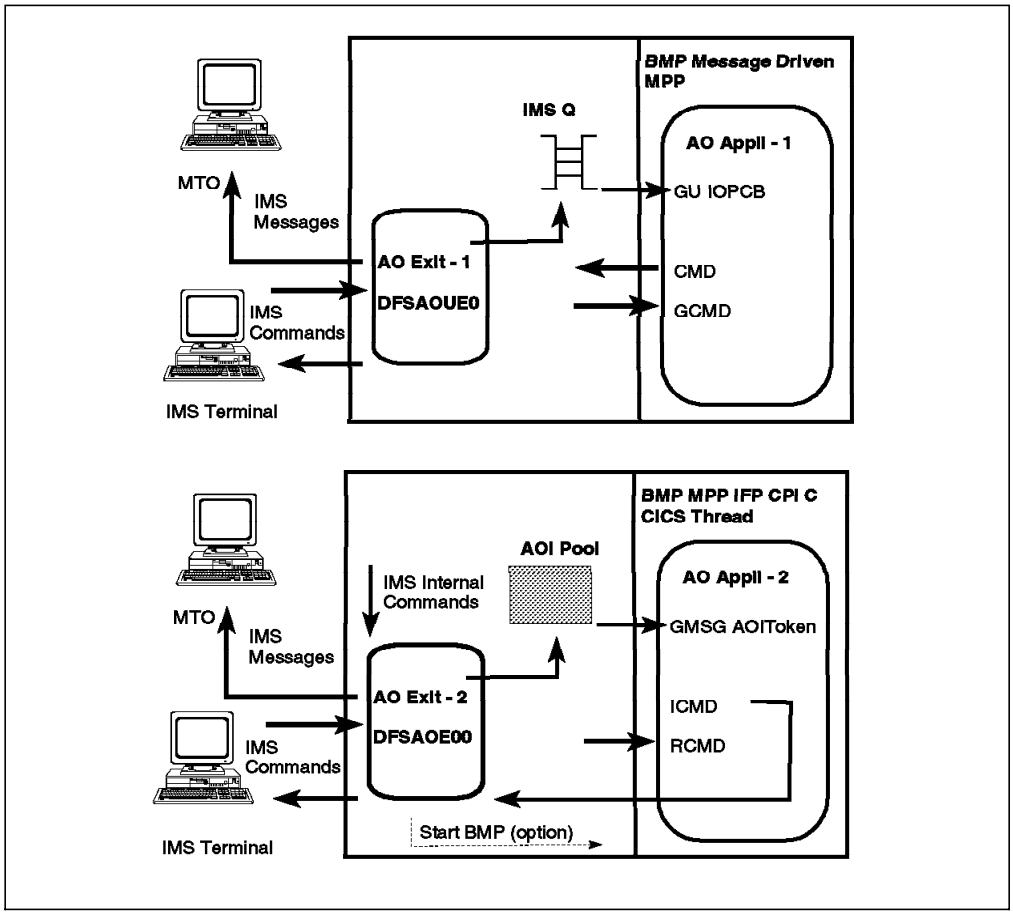


Figure 16. IMS 5.1 Automated Operator Interface

4.1.2 Description

The new (type 2) automated operation facility has greatest value in a DBCTL environment. It can also be used in an online environment. A comparison with the old (type 1) facility is provided in Table 6 on page 72.

Automated Operator Exit - Type 2 (DFSAOE00)

Functionally, the new AO exit is very similar to the DFSAOUE0 exit. It examines IMS system messages, commands and command responses, manipulates them, and generates messages, if necessary, for an AO application to process.

Note: A system message is any IMS message (usually containing a message number starting with DFS) that is not a synchronous response to a command.

The exit is fully documented in the *IMS/ESA V5 Customization Guide, SC26-8020*.

When DFSAOE00 Is Called: IMS calls the Automated Operator Exit - Type 2 (DFSAOE00) for:

- IMS system messages destined for the master terminal
- Commands entered *from a terminal* and responses to those commands

  The following commands are exceptions in all IMS environments:

  – /FORMAT
  – /LOOPTEST
  – /MSVERIFY
  – /RELEASE
  – /NRESTART
  – /ERESTART

  The following commands are exceptions in a DBCTL environment:

  – /FORMAT
  – /LOOPTEST
  – /MSVERIFY
  – /RELEASE
- Commands issued *by an AO application* using the new ICMD DL/1 call, and responses to those commands
- IMS internal commands.[6]

Note: Interception of application issued commands and internal commands is a unique feature of the AO Exit Type 2.

IMS does *not* call the Automated Operator Exit - Type 2 (DFSAOE00) for:

- Responses to IMS internal commands
- Responses to commands issued by an AO application using the ICMD call
- Commands issued by an AO application using the CMD call
- IMS system messages not destined for the master terminal.

If both DFSAOUE0 and DFSAOE00 are loaded, DFSAOE00 is called first. DFSAOE00 can process the message or command, or pass control to DFSAOUE0 to do the processing.

[6] For example, IMS will issue an internal /STOP PSB command if an application abends.

What DFSAOE00 Can Do: This exit can:

- Modify the text of IMS system messages.
- Delete IMS system messages BUT NOT commands or command responses
- Direct IMS system messages, a copy of a command, or a copy of the response to an AO application

  For the case of IMS system messages, the original message will be sent to the secondary master and/or MVS console as applicable. Only the master terminal copy is available for editing. For DBCTL, there is only one message destination, and it receives the edited version (or nothing if message is deleted). Consequently, for DBCTL, the original version of the message is lost.
- Start a BMP and override the APARM parameter to pass information to that BMP (the INQY ENVIRON DL/1 call is used to retrieve the APARM parameter).

  This BMP can be either message-driven or non-message-driven (a non-message-driven BMP can be used in a DBCTL environment).

  This BMP can be an AO application.

The Automated Operator Exit - Type 2 (DFSAOE00) uses new AOI Callable Services to INSERT, ENQUEUE, or CANCEL messages in the message buffers. The destination of the messages is an AOI token to which the message is enqueued. AOI tokens are names, *chosen at implementation time*, to identify particular AO applications. Multiple AOI tokens can be specified, in which case multiple copies of the message will be queued for different applications to process. In comparison, for the Automated Operator Exit - Type 1 (DFSAOUE0), the destination is an LTERM or a transaction, and the message flows through the message queues.

A sample exit routine is provided in the IMS.SVSOURCE library.

Automated Operator Application - Type 2

An AO application - Type 1 can retrieve messages from the message queue by using GU/GN calls and issue some commands by using CMD and GCMD calls. It is applicable only in an IMS TM (DB/DC and DCCTL) environment.

An AO application - Type 2 can retrieve messages from the Automated Operator Exit - Type 2 (DFSAOE00) using the GMSG call, issue some of the IMS commands, and look at their responses using the ICMD and RCMD calls.

Type-2 AO applications can be used in both an IMS TM and a DBCTL environment. GMSG, ICMD, and RCMD calls can be issued by MPP, BMP (MD or NMD) and IFP regions and by DRA threads and CPI-C driven applications (after a successful APSB call). They must use the AIB interface.

GMSG Call: The GMSG DL/1 call, issued through the AIB interface, is used to retrieve messages created by DFSAOE00. The application specifies an 8-byte AOI token as a parameter for the GMSG call, and IMS will only return messages associated with that AOI token. For multisegment messages, only the first GMSG call uses a token. Subsequent segments are retrieved with a GMSG call with a blank token specified. The next GMSG call with an AOI token will delete any remaining segments from the previous GMSG call.

By using different AOI tokens, the AO exit routine can direct messages to different AO applications.

A WAIT option is available to make the AO application wait until the next message is written by DFSAOE00. This option is selected by the application, on a call-by-call basis. The /PSTOP command is used to post an AO application that is waiting on a specific AOI token.

ICMD and RCMD Calls: The DL/1 ICMD call issues an IMS command and retrieves the first segment of the response if one exists. RCMD calls are used to retrieve the second and subsequent segments of a multisegment response. An ICMD call will delete any remaining response segments from the previous ICMD call.

A list of the supported commands can be found in the *IMS/ESA V5 Operator's Reference, SC26-8030* in the "IMS Command Language" section.

The following commands and keywords are not allowed:

- /CANCEL
- /CHECKPOINT with ABDUMP, DUMPQ, FREEZE, PURGE, and QUIESCE keywords
- /END with no keywords
- /ERESTART
- /EXCLUSIVE with no keywords
- /EXIT with no keywords
- /HOLD
- /IAM
- /LOCK with LTERM, NODE, and PTERM keywords
- /MODIFY
- /MSVERIFY
- /NRESTART
- /RCLSDST
- /RCOMPT
- /RELEASE
- /RESET
- /SET
- /SIGN
- /UNLOCK with LTERM, NODE, PTERM, and SYSTEM keywords.

Table 6 on page 72 summarizes the differences between the type 1 and type 2 AOIs.

Table 6. AOI Facilities: Types 1 and 2		
	Type 1	**Type 2**
AO Exit		
Name	DFSAOUE0	DFSAOE00
Environment	IMS TM	DBCTL + IMS TM
To send message to AO Application	Contents of registers	AO callable services
Processes commands from	• Terminal	• Terminal • Type 2 application using ICMD • IMS internal
Edited command buffer visible	Yes	No
AO Application		
Get first segment	GU	GMSG with token
Get next segment	GN	GMSG without token
Issue command and get first response	CMD	ICMD
Get next response	GCMD	RCMD
Command security	SMU	RACF and/or DFSCCMD0
Usable when message queues are unavailable	No	Yes

AOI Callable Services

IMS 4.1 introduced callable services for user exits to use. Two types of services could be requested: storage services and control block services.

In IMS 5.1, a third type of service is added, AOI services. AOI services support three functions:

- Insert message

  The INSERT function inserts a message segment into a message buffer. The message segments are not available to the AO application until the ENQUEUE function has been used to specify the AOI token.

- Enqueue message

  The ENQUEUE function inserts the last or only message segment into a message buffer. It enqueues this message to the AOI token (or tokens) specified by the exit in the token list.

  An AO application can then issue a GMSG call, specifying the AOI token, to retrieve this message.

- Cancel message

  The CANCEL function cancels messages that are not yet enqueued to an AOI token.

Refer to the *IMS/ESA V5 Customization Guide, SC26-8020* for full information on using AOI Callable Services.

4.1.3 Implementing the New Automated Operator Interface

There are some differences in the procedures used with the new (type 2) AOI than with the type 1 AOI in earlier IMS releases. The most important differences affect security and restart message processing.

Security Considerations

Command security controls can be performed by using RACF (or any equivalent security product) and/or the Command Authorization Exit (DFSCCMD0). The Command Authorization Exit (DFSCCMD0) lets you secure commands issued by the ICMD call at the command verb, keyword, and/or resource name level. To choose the security method, specify the new AOIS parameter in the IMS procedure.

This is a significant improvement because the CMD call only uses the Security Maintenance Utility (SMU) for security verification.

The *IMS/ESA V5 Administration Guide: System, SC26-8013* contains a chapter on ″Establishing IMS Security″ with information on the use of AO application programs.

Restart and Recovery Considerations

Because the AO Exit - Type 2 does not use the IMS message queues, messages directed to an AO application cannot be recovered on restart. The AO application can issue the appropriate /DISPLAY commands to determine whether an earlier ICMD call executed successfully.

Startup Parameters

The AOIS parameter is used to request command security in an AO application.

The AOIP parameter specifies an upper expansion limit for the AOI buffer pool.

IMS Commands

A new keyword, AOITOKEN, can be used in the following commands:

- /DEQUEUE command, to dequeue and discard messages associated with an AOI token name
- /DISPLAY command, to show all the AOI tokens in the system
- /PSTOP REGION command, to post an AO application waiting for messages associated with an AOI token. AIB return code X′00000004′ and reason code X′0000004C′ are sent to the AO application.

A status of “WAIT-AOI” in response to a /DISPLAY ACTIVE command indicates that a region contains an AO application waiting on a GMSG call.

The /DISPLAY POOL shows the storage usage in the Automated Operator Interface Pool (AOIP).

4.1.4 Migration from Type 1 to Type 2 Automated Operator Interface

Migration from Automated Operator Exit - Type 1 (DFSAOUE0) to Automated Operator Exit - Type 2 (DFSAOE00) is not mandatory for IMS DB/DC or DCCTL users. Either or both can be used.

If both user exits exist, the new exit (DFSAOE00) will be called. A standard option of the new user exit is to specify that the current message be passed to the old exit (DFSAOUE0) for processing.

4.1.5 Value

DBCTL users can now automate their IMS operations by using the new AOI facility.

For the IMS TM user, the new AOI has certain advantages over the original AOI:

- It does not use the IMS message queues and is therefore still usable when the queues are not available.
- The Automated Operator Exit - Type 2 (DFSAOE00) can see IMS internal commands and commands submitted through ICMD calls.
- Security is provided by RACF rather than SMU.

4.2 Enhanced Database Commands

These small enhancements in the ability to manipulate databases can have a disproportionately large benefit in many environments. This is particularly true for systems with very large numbers of databases.

4.2.1 Overview

There are many times when the IMS operator needs to perform some function (/DBR, /START, etc) on a group of databases or DEDB areas. This may require that multiple commands be entered. The resulting wait while the commands process can be inconvenient, and the stream of messages while the databases complete processing is generally of no value and might hide a genuinely important message.

All these issues are addressed in IMS 5.1.

4.2.2 Data Groups

The concept of grouping database data sets has been available within DBRC for several years. The group is a *database data set group* and is only usable with DBRC commands. It is intended to facilitate situations where a group of data sets is to be image copied, recovered, or the like using the DBRC GENJCL command.

The new IMS 5.1 facility allows *groups of databases* to be defined, with the objective that the group name can be used in the IMS /DBR, /DBD, /START, and /STOP operator commands. For example:

```
/DBR DATAGROUP(BRANCH3)
/DBD DATAGROUP(ACCOUNTS)
/STA DATAGROUP(PAYROLL)
/STO DATAGROUP(LEDGER)
```

Both types of groups are defined with the same DBRC command, INIT.DBDSGRP. A DBDS group (for DBRC purposes) is defined with:

```
INIT.DBDSGRP  GRPNAME(name)  MEMBER((DBname,DDname),....)
```

A database group (for command purposes) is defined with:

```
INIT.DBDSGRP  GRPNAME(name)  DBGRP(DBname,....)
```

The MEMBER and DBGRP parameters are mutually exclusive, which means that a DBDS group and a database group cannot have the same name. In other words, a particular group of databases is given one name for DBRC command purposes and a different name for MTO commands.

The rules governing the new database group facility are:

- A database group can contain any mixture of:
  - Full function database
  - DEDB
  - DEDB area.
- The same database or area can be defined in multiple groups.
- If a new data set is added to a database, or a new area to a DEDB, it automatically becomes included in the database group.
- The maximum number of databases and/or areas per group is 1024.
- Databases and areas do *not* need to be registered in DBRC.
- The contents of a database group are altered with:

  ```
  CHANGE.DBDSGRP GRPNAME(name) ADDDB(DBname,...) DELDB(DBname,...)
  ```

- A group is deleted with:

  ```
  DELETE.DBDSGRP GRPNAME(name)
  ```

- An IMS command with the DATAGROUP parameter is executed as if a *single* command were used containing all of the database and/or area names.
- An IMS command with the DATAGROUP parameter cannot have the GLOBAL parameter. In other words, DATAGROUP commands are always LOCAL commands. This is because when these commands are used with the GLOBAL parameter, there is a DBRC interaction (for example, setting the "prevent further authorization flag") for each database, and the total DBRC processing for a data group could be undesirable. The significance of this must be appreciated in a data sharing environment.

Implementing Data Groups

The use of INIT.DBDSGRP is all that is needed to define a data group.

However, given the two distinct types of groups, some thought should be given to an appropriate naming convention.

Users of large Fast Path DEDBs might consider defining a data group for each DEDB. The group name can be the DEDB name, and the members are the areas:

```
INIT.DBDGRP GRPNAME(MYDEDB) DBGRP(MYAREA1,MYAREA2,.....)
```

Before IMS 5.1, after a /DBR DATABASE MYDEDB, it was necessary to issue /STA DATABASE MYDEDB followed by typically *multiple* /STA AREA MYAREA1 MYAREA2.... commands.

With the IMS 5.1 data group name equal to the DEDB name, it is possible to replace all of the start area commands with a single /STA DATAGROUP(MYDEDB).

4.2.3 DBALLOC/NODBALLOC

Some database START commands can take an inconveniently long time to process. For example:

```
/START DATABASE ALL
/START DATAGROUP(grpname)
```

A major contributor is the time taken to perform dynamic allocation. For full function databases, the allocation information has to be read in from RESLIB.

In IMS 5.1, the NODBALLOC parameter can be specified on the /START DATABASE and /START DATAGROUP commands. It is *only applicable to full function databases* and is ignored for any DEDBs or areas in a data group.

NODBALLOC causes the allocation and the associated I/Os to be deferred until data set open time and so expedites the command processing. For example:

```
/START DATABASE name1,name2,name3,..... NODBALLOC
/START DATABASE ALL NODBALLOC
/START DATAGROUP(grpname) NODBALLOC
```

The general default is DBALLOC (as in previous releases). However, for /START DATAGROUP and /START DATABASE ALL, the default is NODBALLOC.

4.2.4 Command Message Suppression

Prior to IMS 5.1, a database command containing the ALL parameter would produce a large number of messages to the operator that had little or no value. For example:

```
DFS2500I DATABASE dbname SUCCESSFULLY ALLOCATED
DFS0488I cmd COMMAND COMPLETED. DBN|AREA|ADS=name RC=00
```

IMS 5.1 eliminates these messages for:

```
/START DATABASE ALL       /START DATAGROUP(grpname)
/STOP  DATABASE ALL       /STOP  DATAGROUP(grpname)
/DBD   DATABASE ALL       /DBD   DATAGROUP(grpname)
/DBR   DATABASE ALL       /DBR   DATAGROUP(grpname)
```

Allocation errors or any message with a nonzero return code will not be suppressed.

When all databases have finished their command processing, a single message is sent to the operator:

```
DFS0488I cmd COMMAND COMPLETE. DATAGROUP(grpname)
```

or

```
DFS0488I cmd COMMAND COMPLETE. KEYWORD ALL
```

4.2.5 Command Language Modification Facility

All command keywords and synonyms are defined in the DFSCKWD0 module. The user can edit the source file (in IMS.SVSOURCE) to:

- Change a keyword to a new value
- Add, change or delete synonyms for the keyword
- Specify whether the ALL parameter is allowed.

The ALL parameter has previously been suppressible for commands against DATABASE, LINE, LINK, LTERM, NODE, PROGRAM, PTERM, RTCODE, SUBSYS,

TRANSACTION, and USER. Many customers with large numbers of resources defined to IMS have disabled the ALL parameter for one or more of these keywords.

However, in IMS 5.1, the use of ALL on DATABASE commands no longer has the impact it used to, with perhaps one exception, namely the /DISPLAY command. Consequently, IMS 5.1 allows the ALL parameter to be disabled either for all commands or just for /DISPLAY commands. The new KEYWD macro format is:

KEYWD *keyword*,LAST=NO|YES,ALL=YES|NO|DIS

To allow use of ALL with DATABASE commands except /DISPLAY, the DFSCKWD0 source would be changed to:

```
IKEY     DAT
DFSCMDFL DBTM=Y,DBCT=Y
DFSCMDRL ACTV=Y,XALT=Y,TRKR=Y
KEYWD    DATABASE,ALL=DIS     (ALL NOT ALLOWED ON /DISPLAY)
SYN      DB
SYN      DATABASES
SYN      DBS
```

The specification of ALL=DIS does not apply to /DISPLAY commands submitted through the AOI.

IMS 5.1 extends the ALL=YES|NO|DIS facility to all keywords.

4.2.6 Value

The operator productivity benefit of having to enter one command in place of many is self-evident. Similarly, an automated operator script can be simplified by this facility. Large Fast Path users may find the data group facility particularly welcome when DEDBs contain large numbers of areas.

Operators will appreciate the benefits of commands completing quicker and without an excessive number of messages arriving at the terminal.

4.3 Enhanced DBCTL Operational Interface

IMS 5.1 has responded to customer requirements for an improved operational capability for DBCTL by providing these enhancements, which should make operating DBCTL easier and with less chance of error.

4.3.1 Overview

A DBCTL system has no IMS master terminal, and so operator commands are entered at an MVS console. In IMS 5.1, various enhancements have been made to this operator interface to simplify the entering of commands and reduce the number of lines of command responses.

4.3.2 Message Elimination

In previous releases of IMS/ESA, replies to DBCTL commands were prefixed with an extra line, for example:

```
       DFS000I MESSAGE(S) FROM ID=IMS4   0017
 0017  DFS994I COLD START COMPLETED
```

(The 0017 at the end of the first and the beginning of the second line is the MVS multiline message identifier.)

The first line is only relevant for supplying the IMS ID when there are multiple DBCTL systems on that MVS system.

IMS 5.1 allows the first line to be suppressed and the IMS ID to be included in the remaining message text. For example:

```
DFS994I COLD START COMPLETED IMS4
```

This option is requested with a DBCTL region EXEC parameter:

```
PREMSG=N
```

The old format is still available with PREMSG=Y.

One exception to the prefix message suppression is when a DISPLAY command is used. In this case the displayed output will be prefaced with a DFS4444I message.

4.3.3 Entering DBCTL Commands

Previously, each DBCTL system was allocated a unique command recognition character (CRC). When an IMS command is entered at the MVS console, the CRC identifies which DBCTL system is to process the command.

In IMS 5.1, entering commands is made more flexible.

Option to Eliminate the CRC

It is possible to request that no CRC be allocated for a DBCTL system. This is a system definition option on the IMSCTRL macro:

```
IMSCTRL CMDCHAR=NONE
```

The way to target a command at a particular DBCTL system when there is no CRC is explained below.

DBCTL Command Formats

When a CRC is defined, IMS 5.1 commands can be entered in exactly the same way as in previous releases, or they can use a new command format. When no CRC is defined, the new command format must be used.

Both command formats (old and new) are affected by whether the command is single or multiple segment. Most commands are single segment, but the following are the multisegment commands valid in a DBCTL environment:

```
/CHANGE
/ERE
/RMxxxxxx
/SSR
```

Old Command Format: In this example, the CRC is a question mark (?).

- Single segment command

  The first character of input is the CRC:

  ```
  ?DBR DATABASE ORDERS
  ```

- Multisegment command

All segments but the last must start and end with the CRC. The last segment must start with the CRC:

```
?RMINIT DBRC='IC DBD(DEDB003) AREA (DB03AR05) ICDSN(ICF51.DE03AR05.COPY1) ?
?ICDSN2(ICF51.DE03AR05.COPY2)'
```

New Command Format: The IMS ID of the DBCTL system is used in place of the CRC. In the following examples, DCT4 is the IMS ID:

- Single segment command

  ```
  DCT4DBR DATABASE ORDERS
  ```

- Multisegment command

  ```
  DCT4RMINIT DBRC='IC DBD(DEDB003) AREA (DB03AR05) ICDSN(ICF51.DE03AR05.COPY1) DCT4
  DCT4ICDSN2(ICF51.DE03AR05.COPY2)'
  ```

The old and new formats cannot be mixed in any one multisegment command. Different multisegment commands can be in either format.

Note: For *single segment commands only*, there is an alternative method of entering a command when the IMS ID is less than four characters. The command verb can be started exactly four characters after the start of the IMS ID. For example, if the IMS ID is DC5, the following are both valid:

```
DC5 DBR DATABASE ORDERS
DC5DBR DATABASE ORDERS
```

If the IMS ID is D6, the following are valid:

```
D6  DBR DATABASE ORDERS
D6DBR DATABASE ORDERS
```

The second method, with no blanks between the IMS ID and the command verb, is recommended.

4.3.4 Value

Eliminating the prefix message will reduce the message traffic to the MVS console and so enhance the operating environment.

The changes associated with CRCs will be appreciated by those customers with several DBCTL systems. Some customers have had problems finding enough suitable CRCs for their many systems. For them, the elimination of the CRC will be helpful. But even customers with one or a few DBCTL systems can use either command format. Operators may find it easier to remember an IMS ID than a CRC and will make fewer errors.

Chapter 5. IMS 5.1 Availability and Remote Site Recovery

This chapter describes the IMS Version 5.1 enhancements in the general area of high availability. This includes data, MSC link, and system availability, and in particular, the new Remote Site Recovery facility.

5.1 Remote Site Recovery Feature

The RSR feature of IMS 5.1 is a separately priced component that provides a disaster backup (remote site) capability for an IMS system. It replaces the RRDF product used with IMS 4.1 and provides a very different solution, which is much broader in its scope and capability.

Only IMS databases are protected by RSR. Other system resources, such as MVS catalogs, nondatabase data sets, DB2 tables, and application load module libraries, are not covered by RSR and must be copied to the remote site by other means.

5.1.1 Overview

In the event of an extended outage of an IMS primary system, whether temporary or permanent, planned or unplanned, RSR enables resumption of the IMS service at a remote site within minutes, with assured data integrity and consistency and minimal or no loss of data updates and messages.

The key characteristics of RSR are as follows:

- RSR is used to recover *IMS resources*:
  - IMS Full Function databases
  - IMS Fast Path DEDBs
  - IMS TM message queues.

  It cannot be used to recover CICS, DB2, or other resources.
- RSR transmits to the remote site copies of the IMS log records necessary for database and message queue recovery. There they are received by an IMS RSR tracking system.
- RSR lets the installation select the minimum number of log records that are needed to support the critical business environment.
- RSR allows shadow databases to be maintained at the tracking site. With this option, the updates read from the transmitted log data are applied to the shadow databases as they are received at the tracking site.

  Databases for which shadowing is not requested will need to be recovered at the remote site, before the full service can be resumed after a takeover.

  Shadow databases are not accessible by any program on the tracking site before the takeover.
- RSR does not impact normal IMS processing.
- RSR provides a way of automatically resynchronizing the two sites as soon as possible after a log transmission breakdown. The transmission of gaps in the log sequence is performed by the isolated log sender (ILS) function.

- RSR does not change IMS logging or recovery processing. It just extends recovery to the tracking site, where enhanced recovery facilities are available.
- There are IMS commands to define, update, and display RSR status.
- RSR requires the services of DBRC, which is responsible for:
  - Knowing which databases are associated with a tracking site
  - Keeping information concerning the log data received at the tracking site
  - Assisting at a takeover.
- RSR understands transactions and units of recovery. Recovery at the tracking site (by shadowing or not) maintains database integrity and consistency.
- RSR supports both BLDS and XRF systems.
- RSR uses a closed LU6.2 SNA communications protocol.

Note: Some synonyms:

- Local site = active site = primary site.
- Remote site = tracking site = secondary site

5.1.2 Aspects of IMS Service Recovery

Two types of recovery have to be considered, local recovery and remote recovery.

Local Recovery

For local recovery, several methods can be used to ensure the timely recovery of the computer services. They are discussed later in this section, where their applicability to remote site recovery is considered. Figure 17 on page 83 illustrates some local recovery facilities.

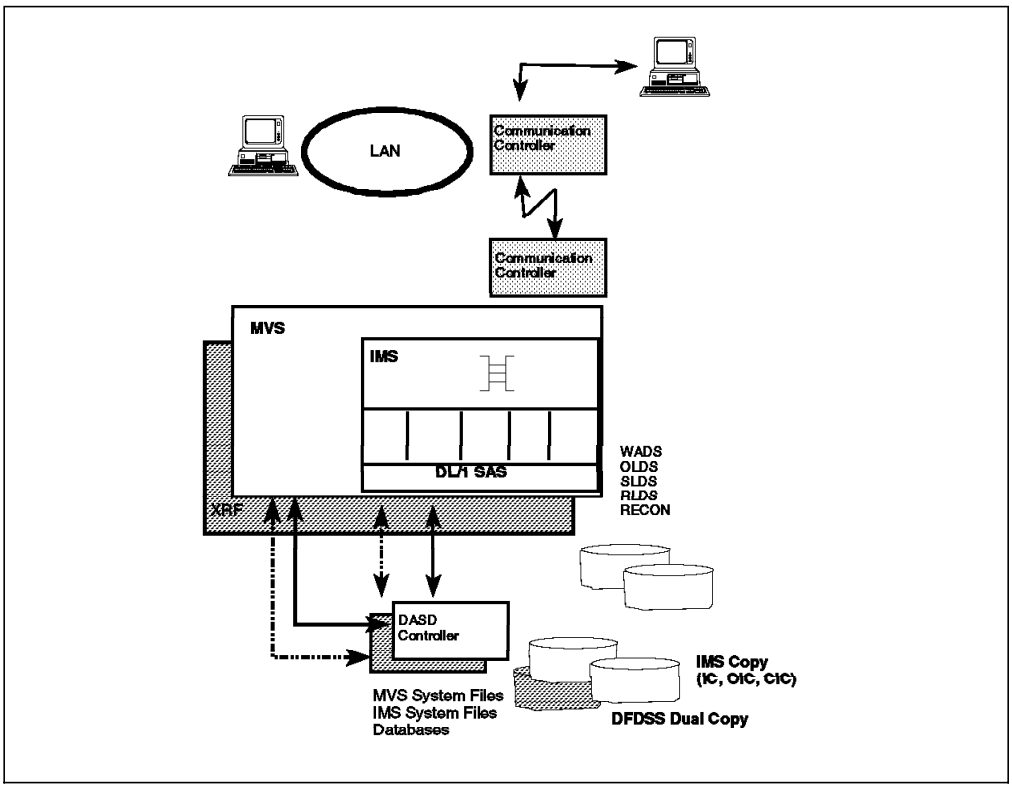


Figure 17. Local Recovery

Remote Recovery

For remote recovery of an IMS service, many customers have adopted a basic principle of making regular copies of the important data and shipping the data to a remote location. They use the traditional methods available in the MVS environment like pack dumps or image copies.

Some customers have tried to improve upon this, especially for their databases, by additionally shipping archived log data at regular intervals. The remote site can either be a partition of another MVS system within the same customer enterprise, or it can be rented from another company that provides contingency facilities. The site can vary from an empty computer room (equipment is installed only if a disaster strikes) to a partition on an existing system with regularly recovered databases.

Components of an MVS System

An MVS computing service is provided by a system with many components. These components have different recovery techniques, depending on the nature of the data and its volatility, the availability of a recoverable log, the need for integrity and consistency, and the cost effectiveness of the various alternatives.

For an IMS service, the main components are:

- MVS system and files
- IMS system and libraries
- IMS logs and log control data sets
- IMS databases
- IMS message queues
- Hardware

  - CPU
  - DASD controller
  - DASD
  - Communications controller
  - Communications links
- Miscellaneous
  - People
  - Supplies
  - Automation.

When thinking about remote recovery for these various components, it is important to keep in mind the costs associated with data and service loss, compared with the costs of minimizing those losses.

5.1.3 Component Recovery (Software Solutions)

MVS System and Files

This part is relatively static. The system and the system files do not change as often as the contents of the databases.

Local Recovery: For local recovery, the need to have another MVS system available with copies of the system files is obvious.

Remote Recovery: For remote recovery, an MVS system needs to be operational on the remote site, but if this system is not the latest image, some problems may occur. For example:

- Security violations may occur if the RACF data set is not up-to-date.
- Job scheduling scripts may have been updated.
- Tuning parameters may have changed.

IMS System

Local Recovery: IMS automatically performs recovery processing after errors in the transaction or database managers, application programs, or databases.

For local recovery, the IMS system provides various facilities:

- System and application checkpoints
- Emergency restart capability
- Backup copies of IMS TM message queues
- Dual option for log management data sets
- Automatic database backout
- Utilities to assist in the recovery of the log
- Utilities to assist in the recovery of databases.

IMS also provides a hot stand-by solution for local recovery, namely the **XRF**. XRF provides continuous availability of the IMS database and transaction managers within a single complex. An alternate system continuously monitors the active system and watches for a failure of the IMS service. When a failure occurs, MVS, VTAM, and the NCP work with IMS to enable the alternate system to take over the IMS workload with minimal disruption.

Because the databases, OLDS, and RECON DASD must be shared, the distance between the two systems is limited by ESCON channel distances.

XRF is still usable in IMS 5.1 for providing local site recovery and is complemented by RSR to provide remote site recovery.

Remote Recovery: This is essentially the same as for the MVS system. The customer has to ensure that an up-to-date copy of the IMS active system (or a subset) is available at the remote site.

In IMS 5.1, the RSR facility is provided by having a remote IMS system that tracks the active IMS system′s updates. At a takeover, this tracking system is stopped and a copy of the original IMS is started. It is the customer′s responsibility to provide the remote IMS system.

IMS Libraries

A number of libraries are critical to the IMS operation, for example, RESLIB, MODBLKS, MODSTAT, ACBLIB, MFS library and application PGMLIBs.

Local Recovery: The customer is responsible for ensuring that these libraries can be easily recovered.

Remote Recovery: Again, the customer is responsible for making them recoverable, and for keeping the local and remote copies the same. This is especially critical for libraries that potentially change regularly, namely ACBLIB, MFSLIB, and PGMLIB.

If online change is used, the other libraries also need to be brought up-to-date at the remote site.

With RSR, IMS provides the benefit of tracking the MODSTAT data sets.

WADS, OLDS, SLDS, RLDS, and RECON

Local Recovery: Protection of these vital resources is provided by replication. It is normal for IMS to use dual logging for all of its log data sets.

DBRC also needs to have three RECON data sets (two actives and a spare). Those should be of different size and can be put under different DASD controllers to maximize the DBRC information availability.

Remote Recovery: Many customers have made some attempt to protect themselves from a disaster. Some simply use manual or semiautomated procedures for shipping archived logs to an offsite location. Some customers have developed electronic tools to transfer the log data.

A few customers are using a separate product, Remote Recovery Data Facility (RRDF) from the ENET Corporation, to transfer IMS log data and DBRC information to a remote site.

Now as a feature of IMS 5.1, RSR provides the same facilities but enables a simpler environment to be run and managed and provides the additional benefit of a remote database shadowing facility.

Databases

Local Recovery: Image copies and logs are used on the local site to rebuild databases that are broken or contain inconsistent data. DEDB databases provide an alternative way of improving availability by using area replication. This function is called Multiple Area Data Set (MADS).

Remote Recovery: As with log shipping, customers use a range of procedures for providing copies of databases at the remote site.

Even customers who have not ventured to ship log data offsite tend to make some effort to provide copies of databases for use after a disaster. Clearly, without the appropriate log data, weekly or even daily database backup leaves enterprises at risk of losing all subsequent updates. The database can be recovered only to the timestamp of the last image copy that is available and readable at the backup site. Because of tape errors, bear in mind that if you take copies every n days, you have to be prepared to lose 2n days worth of data (or more).

The quantity of lost updates can be reduced by providing the IMS logs at the remote site and running database forward recovery. If archived logs are transferred, the updates at risk of loss are those on the current online log, together with those on logs still in transit. Unless electronic log transfer is used, there will be a significant delay before recovery can be performed.

With RRDF and RSR, the log records are shipped when they are written to the active OLDS, and so the number of lost updates is kept to a minimum (and is zero in a scheduled takeover). The advantage of RSR is that the log records can be applied to the shadow databases immediately after they are received at the remote site. There is no need to wait for the remote site archive to take place or run a separate recovery job for each database. In other words, RSR minimizes both the amount of data lost and the time taken to restart the IMS service at the remote site.

Message Queues

Local Recovery: Every input or output message is recorded on the IMS log.

Normally, IMS itself recovers IMS message queues by processing the IMS log. The IBM Message Requeuer (product number 5655-038) can be used to rebuild the message queues from the log after a cold start.

Remote Recovery: With RSR, the queues can be rebuilt by the remote IMS as it starts or can be built by the MRQ product. In other words, it is just like the local system.

5.1.4 Hardware Solutions

Local Recovery

For local recovery, it is possible to have a configuration with duplexed DASD on a 3990 DASD controller using the **Dual Copy** feature. This facility is transparent to MVS and therefore also to IMS.

It is also possible to have two DASD controllers, with duplexed channel paths, accessing the same DASD. In case of a DASD controller problem, the second channel can still be used.

Remote Recovery

For remote recovery, the ESCON technology associated with the 3990 model 6 DASD controller provides the option of doing **remote copy**. A second copy of selected DASD volumes can be made synchronously through ESCON technology (logical path, directors, and extenders) or asynchronously through any type of channel. Remote copy supports all DASD data, such as databases, RECONs, and logs in the IMS environment, and MVS system files or the RACF data set in the MVS environment.

Peer to Peer Remote Copy (PPRC) is the synchronous option, managed by direct communication between the two 3990 model 6 controllers connected by an ESCON channel.

Extended Remote Copy (XRC) is the asynchronous option and uses a data mover application on a different MVS system to send the updates from a 3990 model 6 to a 3990 model 3 or 6.

Figure 18 summarizes the component software and hardware involved in providing remote copy facilities.

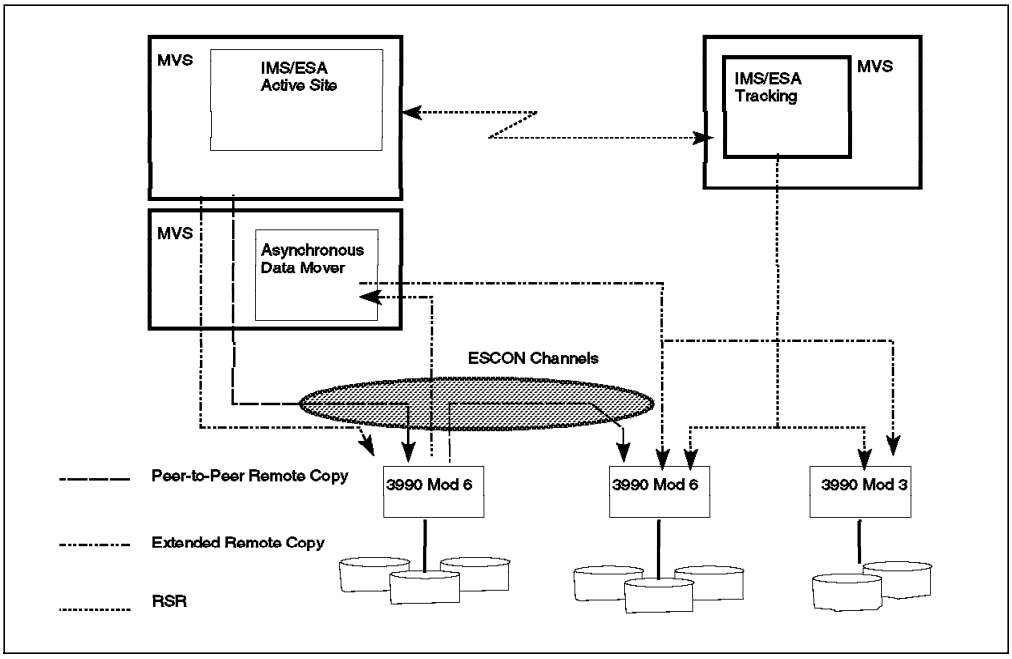


Figure 18. Remote Recovery

The remote DASD solutions are limited both by ESCON channel distances and the propagation delay that will exist if the copy is synchronous. If asynchronous copy is used, some incoherence can be introduced in the remote databases after a failure.

The network cost has also to be evaluated. *All write I/Os* to every duplexed DASD volume have to be transmitted to the remote site.

The remote copy process is independent of IMS and has no understanding of transactions and units of recovery. Where data integrity and consistency are important, the use of remote copy for databases and logs must be carefully assessed.

For ICs, OLDS, WADS, SLDS, RECON, and MADS, one copy can be local, and one copy can be remote.

The remote copy facility can be used to maintain copies of the IMS libraries, (for example, RESLIB, MODBLKS, ACBLIB, MFS library, and application PGMLIBs) at the remote site. Good candidates are files that are not often modified and for which updates do not impact IMS processing.

Network Hardware

The network specialists have to think about switching between sites, replication of network components, and the implications for communication controllers, links, and VTAM and NCP parameters. In the case of a transfer of service from a local to a remote site, VTAM and NCP should be able to dynamically change the routing for the terminals.

5.1.5 Inside the Scope of Remote Site Recovery

The RSR feature provided in IMS 5.1 supports IMS DB/DC, DBCTL, and IMS Batch environments. In the case of a scheduled takeover, no updates will be lost. In an unscheduled takeover, RSR minimizes the number of lost updates.

The time taken to restore the service after a takeover depends primarily on whether database shadowing has been adopted or not. With shadow databases, the service recovery time can be just a few minutes. If some or all databases are not shadowed, then clearly they must first be recovered before the full IMS service can be restored.

RSR supports the following:

- Remote recovery of databases using the following access methods:
  - HDAM, HIDAM, HISAM, and SHISAM.
  - Fast Path DEDB including VSO areas.

  MSDBs and GSAM sequential files are not supported.
- MADS for the shadow copies of the DEDB areas. The number of area data sets for each area at the remote site can be different from the number at the active site.
- Online change, by tracking the MODSTAT information. One MODSTAT data set must be defined for the tracking subsystem itself. Then one additional MODSTAT data set must be defined for each MODSTAT data set that has to be tracked. It indicates the current ACB, FORMAT, MATRIX, and MODBLKS libraries in use.

The installation decides the subset of databases that have to be tracked and registers them in DBRC with the RCVTRACK or DBTRACK parameter. The remote RECON contains only information about the “tracked” databases. Each installation must ensure that a coherent set of databases is tracked, for example, all logically related databases, databases and all their indexes, or all databases relating to a single application.

5.1.6 Outside the Scope of Remote Site Recovery

The RSR feature only takes account of IMS logs and databases. Other facilities used by the IMS system require alternative techniques. This includes IMS libraries, MVS data sets (such as RACF data sets), and application files (for example, those used in BMPs with GSAM). For some of these, as has already been mentioned, the hardware remote copy facilities may be an affordable choice.

Hardware Resources

The hardware resources required at the remote site depend on the proportion of the active work that is tracked. Clearly, there must be enough resource to run the critical workload. However, not all of this resource is needed while in tracking mode. Before the takeover, the tracking site processor should be large enough to support the RSR workload, which depends on the chosen readiness level of recovery (see “Readiness Level of Recovery” on page 90).

The DASD and tape devices do not need to be the same at the two sites. But the database definitions and physical database characteristics must be the same.

IMS Resources

Any changes to system definitions and system or application files must be replicated at the remote site. This includes specifically updates of ACBLIBx, DBDLIB, FORMATx, JOBS, MATRIXx, MODBLKSx, PGMLIB, PROCLIB, PSBLIB, RESLIB, and TFORMATx. Only changes at the active site to the MODSTAT data set are recognized at the tracking site.

The RECON data set created by RSR on the tracking site is neither a physical copy nor a logical copy of the active system′s RECON data set. It contains information about the tracked databases, the log data received, and the log generated by the tracking system itself.

The data set names at the tracking site must correspond with the names used in the JCL procedures, in the DBRC definitions, and in the dynamic allocation members.

Any alterations to the active system to tune or enhance the operational characteristics should be considered for replication at the tracking site.

5.1.7 Remote Site Recovery Terminology

The RSR Complex

A fully qualified name of a component in an RSR environment is:

```
GSGname.SGname.SYSTEMname.INSTANCEname.COMPONENTname
```

These qualifiers are explained below.

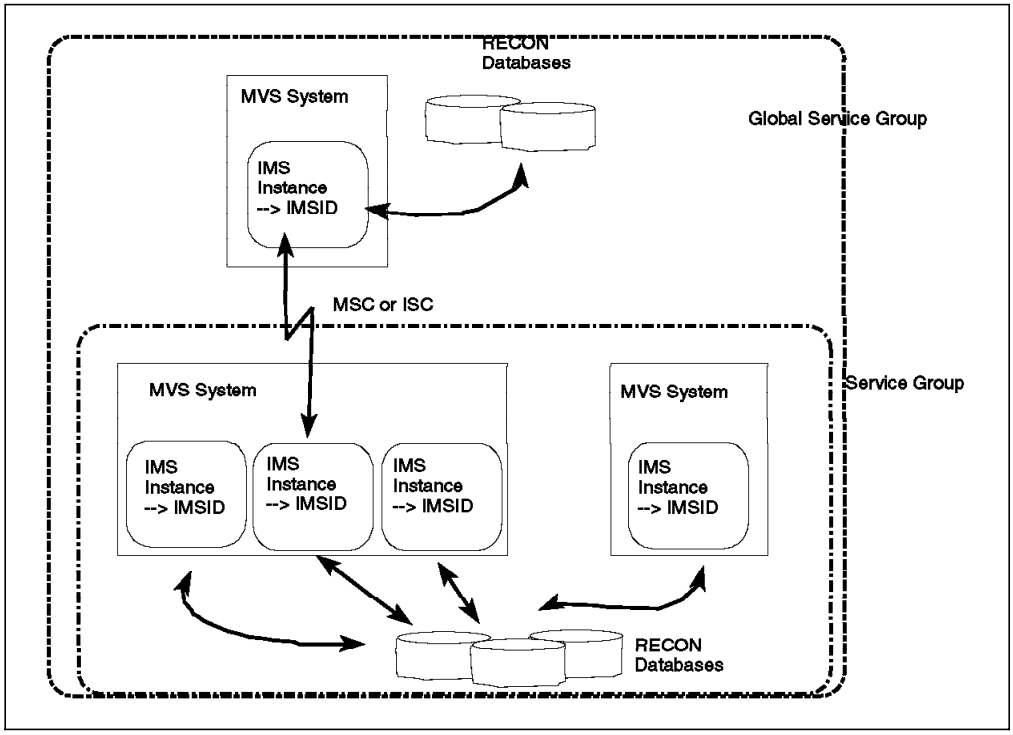


Figure 19. RSR Terminology

Global Service Group (GSG): Collection of all IMS subsystems that access a particular set of databases.

Systems belonging to different GSGs can be connected using MSC or ISC links.

Service Group (SG): The set of databases registered in a single RECON, together with all of the IMS systems that have access to those databases and share the RECON. This includes an XRF alternate system and all data sharing systems.

An RSR active system is one SG, and the tracking system is another.

An RSR GSG is made up of two SGs, the active and the tracker. A remote takeover is a takeover of a service group.

System: An MVS system that has (at least) one instance of the RSR Transport Management Subsystem (TMS) running on it.

Instance: A particular IMS subsystem within the MVS system (DBDC, DBCTL, DCCTL, IMS Batch job or batch utility job) where there is a logger function.

RSR Component: A part of RSR within a particular IMS instance; for example, a logger, a log router, or an isolated log sender.

Readiness Level of Recovery

Two options for tracking are available. The choice is determined by the length of time without IMS service that the customer can tolerate.

An IMS startup option specifies the tracking system′s readiness level. To be managed by RSR, databases must be registered in DBRC with the parameter

RCVTRACK (for recovery readiness tracking) or DBTRACK (for database readiness tracking).

Recovery Level Tracking (RLT): For a database defined with the RCVTRACK parameter, the related log data is sent from the active IMS system to the tracker, where it is archived and kept until recovery or takeover is required.

To minimize recovery times, the customer might choose to run periodic (for example, daily) recovery using these logs. Alternatively, the logs may be used, together with the most recent image copy, only in the event of a takeover being necessary.

Database Level Tracking (DBT): For a database defined with the DBTRACK parameter, the related log data is sent from the active IMS system to the tracker, where it is archived. But it is also sent to the database tracker which updates the shadow database.

On the tracking IMS subsystem, these databases must be registered in DBRC, and IMS processes them as shadow databases. Because the active and shadow databases are updated asynchronously, the most recent updates might not be present on the shadow databases after an unscheduled takeover.

5.1.8 Active Site Components

Figure 20 shows the IMS and RSR components involved in the remote recovery process.

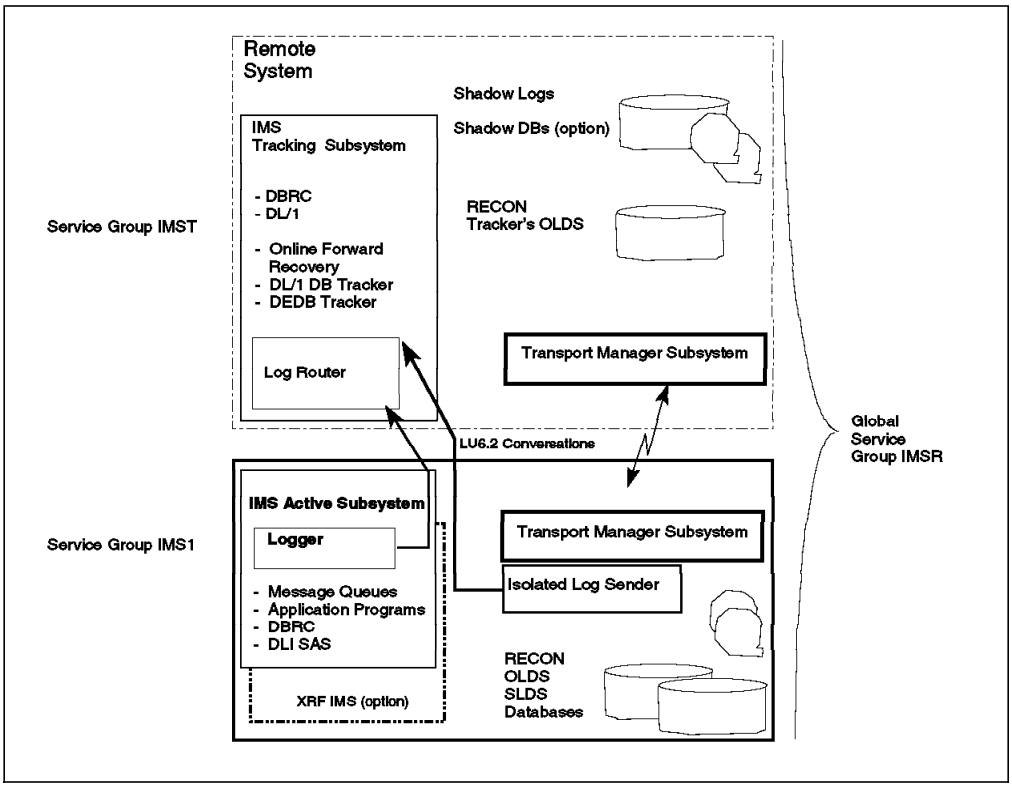


Figure 20. RSR Components

Transport Management Subsystem

The component that provides and manages the communication between the IMS active subsystem and the IMS tracking subsystem is the TMS. It is independent of IMS, running in its own address space, but the IMS subsystems talk to it when they want to participate in RSR. There is one TMS component on each partner and they communicate with each other using two LU6.2 conversations. The TMS is a long-running subsystem under MVS and should be stopped only when MVS stops. When an active IMS starts, it attempts to make contact with its local TMS. A TMS will also have an associated DBRC instance to access the RECON data set.

Five commands are available to operate the TMS: DEFINE, DISPLAY, SET, START, and STOP. They can be invoked from an MVS console using the MVS MODIFY command, or from a SYSIN data set at startup time.

IMS Logger

This is the only IMS component in the active system that is involved in RSR processing.

The IMS logger establishes a conversation with the log router component of the tracking subsystem, with the assistance of the TMS.

Before each OLDS write I/O is issued, the log buffer is sent to the tracking subsystem, where it is stored on a shadow log (SLDS) and optionally routed to the database trackers.

If the communications link between the active and tracking systems becomes temporarily overloaded, some blocks may be missed. The missing log data will be requested asynchronously by the tracking system. If the communication link fails, the sending of log data is suspended until the next OLDS switch. At each log switch, the logger tries to reestablish a conversation with the log router. The IMS /STOP SERVGRP command can be used to stop this process. If communication is reestablished, normal log transfer will be resumed, and the ILS will be requested to send the missing log data.

Isolated Log Sender

The ILS is a component on the active site and runs in the TMS address space. It is responsible for sending log records that could not be sent as they were created. There are various reasons why this may happen, for example:

- TMS is not active at IMS startup.
- A shadow log copy is unreadable when needed.
- A link failure occurs between the two TMSs.
- The link bandwidth is insufficient for the peak logging rate.

These sets of missing log records are called gaps and are recognized by the tracking system because of a break in the log sequence number (LSN). Information about the gaps is stored in DBRC on the tracking site.

The log router on the tracking site detects the gaps and asks the ILS to send the missing log records. Only one ILS is needed in a service group. It will send data from closed OLDS or SLDS. If the gaps cover several SLDSs, they can be filled in parallel using concurrent conversations between the ILS and the log router.

5.1.9 Tracking Site Component

IMS Tracking Subsystem

This IMS system is very different from the active IMS:

- The tracker uses a control region, a DBRC region, and a region for the TMS.
- A DLI separate address space must be present if database level tracking is used for DLI databases.
- The tracker cannot have its own local databases.
- The tracker does not support transaction processing while running in tracking mode. No dependent regions are available.
- The tracker supports a subset of IMS utilities.
- The log router runs in the control region. It creates tracking or shadow logs that contain the log records received.
- The tracker uses its own log data sets (tracker's OLDS in Figure 20 on page 91 and tracker's SLDS for the archived OLDS) to record information required for its own restart and recovery processes. DBRC records this information as well.

Log Router

The log router component receives data from the active subsystems, stores the log data in tracking log data sets, and routes log records to the individual trackers that update the shadow databases. It acts as the source of log data for the other components. The functions that it performs are as follows:

- The log router initiates LU6.2 conversations with the IMS loggers of the IMS subsystems of the active service group.
- It receives the log data from each active IMS logger and creates shadow logs (SLDS-like data sets) on the tracking site. It informs DBRC of SLDS data set creation and all the log data set to database relationships. The log router can automatically archive the closed SLDS. It does not use the IMS archive utility (DFSUARC0). After successful archive, the original SLDS is deleted and the archived copy is the one available for the online forward recovery (OFR) or catch-up processing.

  If more than one active logger sends data to the tracking subsystem, data reception and SLDS creation proceed in parallel.
- When the active subsystem has been unable to send the log data, the log router recognizes a gap in the received log data, contacts the ILS at the active site, and requests the missing log data.
- If DLT is used, log records for the databases involved are presented to the database trackers after they have been written to the SLDS.

  When gaps are detected, no log records beyond those gaps are sent to the database trackers. A procedure called *catch-up processing* is used to present the gap log data to the database trackers.
- OFR is used to recover shadow databases that have been unavailable for a period of time or have had to be initialized from an Image Copy.
- The log router keeps track of control information on the IMS tracking subsystem log (OLDS).

DLI Tracker

The DLI database tracker receives the active IMS log data from the log router and updates shadow databases. It requires exclusive authorization on the database and checks with DBRC to determine whether updates can be applied. No other IMS subsystem or database utilities can access the database until the tracker relinquishes its authorization (/DBR command on the IMS tracker). Even then, DBRC will only authorize Image Copy and recovery utilities (until after a takeover).

The DLI database tracker exists only when the readiness level of recovery is DLT and the databases are defined with DBTRACK.

Fast Path DEDB Tracker

The Fast Path database tracker receives the active IMS log data from the log router. The database updates are:

1. Held in temporary storage until a commit log record appears
2. Ordered by RBA and stored in an MVS data space
3. Accumulated in the MVS data space if several updates occur for the same CI
4. Written to DASD when a data space threshold value is exceeded or periodically based on time or log record count.

The Fast Path database tracker exists only when the readiness level of recovery is DLT and the DEDB areas are defined as DBTRACK.

Online Forward Recovery Utility

OFR is used to recover databases or areas in two cases:

- A shadow database has been unavailable for a period of time.
- A shadow database has been reinitialized from a backup copy.

The utility brings shadow databases or areas up to the current tracking state.

OFR does not use Image Copy and change accumulation data sets. Instead, OFR uses the log data on the shadow logs and passes the updates to the database trackers. Several logs may have to be applied before normal database tracking can be resumed. If an Image Copy is used, the utility automatically excludes the log data created before the Image Copy.

Executing OFR, called **catch-up processing**, is needed in the following situations:

- Network disruptions or network overload
- Failure of a tracking subsystem
- Following a DLI database reorganization at the active site.

OFR is used only for databases on subsystems defined with DLT.

This new utility will recover multiple databases with a single pass of the log and so reduces the service recovery time.

5.1.10 Examples of Tracking Flow

This section illustrates the concepts introduced in the preceding sections, by showing the steps involved for normal tracking and for the case when tracking is interrupted for some reason. Figure 21 shows the system components.

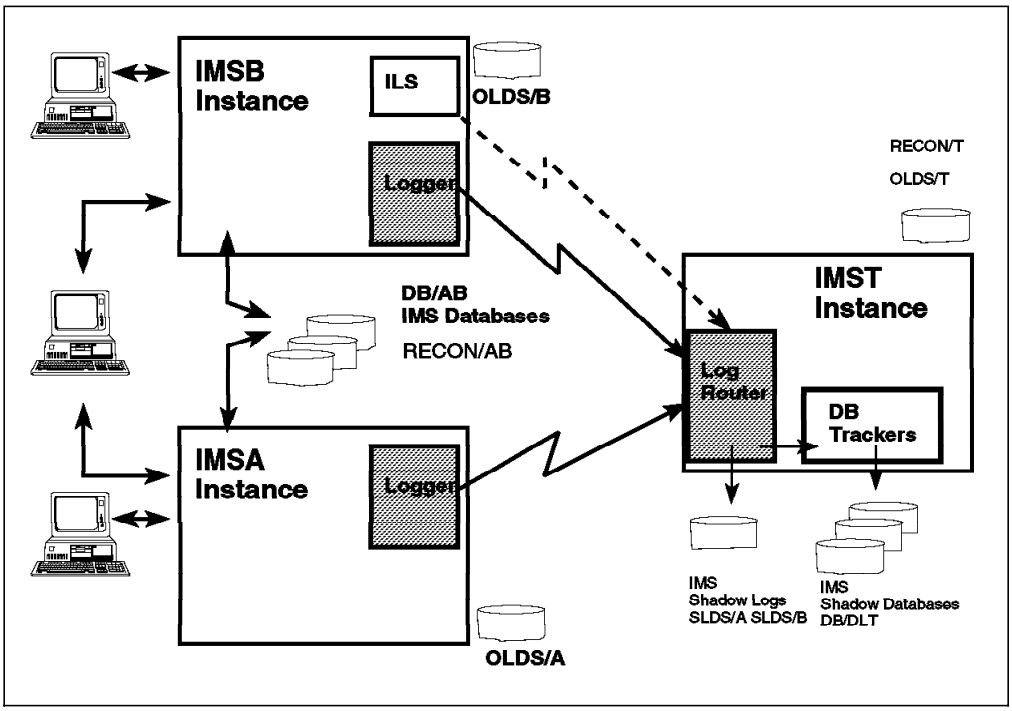


Figure 21. Sample Tracking Configuration

All of these procedures are detailed in the *IMS/ESA V5 Operations Guide, SC26-8029*.

Configuration

IMS systems, IMSA and IMSB, are involved in data sharing on two separate MVS systems. Table 7 lists the abbreviations for tracking flow.

Table 7. Abbreviations for Tracking Flow

	Active IMS A	Active IMS B	Tracking Site
IMSID	IMSA	IMSB	IMST
RECON data set	RECON/AB	RECON/AB	RECON/T
OLDS	OLDS/A	OLDS/B	OLDS/T
Transmitted log data			SLDS/T
Message queues	MQA	MQB	
Shadow database	DB/AB	DB/AB	DB/DLT
Shadow logs			SLDS/A SLDS/B

RECON/AB contains the DBRC log and recovery information for the IMS data sharing environment (IMSA and IMSB).

RECON/T contains the DBRC log and recovery information for the IMS tracking subsystem.

Normal Flow

1. IMSA creates OLDS/A.
2. IMSB creates OLDS/B.
3. Log buffers from IMSA are sent to IMST and stored in SLDS/A.

   The Log Filter Exit (DFSFTX0) can be used to control the amount of log data sent.
4. Log buffers from IMSB are sent to IMST and stored in SLDS/B.
5. RECON/T is updated with information on the log data received.
6. Database changes are applied to DB/DLT as the log records are received from the log router.

Interrupted Flow

Consider the case of a break in communication between the active and tracking systems.

- The log buffers from IMSA are not sent to IMST because the communication has stopped (IMSA operation continues without interruption).

  Database DB/DLT is no longer updated because no log data is being received.

  The RECON/T keeps track of the last log data received from IMSA.
- The log buffers from IMSB continue to be sent to IMST.
- When communication is reestablished:

  1. The IMS tracker recognizes that there is a gap in the log.
  2. The log router requests that the missing log data be sent from the active IMS system.
  3. The log router initiates an LU6.2 conversation with the ILS.
  4. The ILS is activated to transfer the missing log data to the tracker and is then deactivated.

     Until the ILS fills the gap, all subsequent log data is simply written to the SLDS/A and SLDS/B without being passed to the database trackers. This creates “marooned” log data.
  5. As the gap is filled, SLDS/A is created and the log data is presented by the OFR to the database trackers to update DB/DLT.
  6. Catch-up processing will continue by updating DB/DLT with the marooned log data.
  7. Any log data written to SLDS/A after catch-up processing began will also be passed to the database trackers, until the catch-up processing has caught up with current tracking.
  8. Normal tracking is then resumed.

Planned Takeover

A planned takeover allows the transfer of all work from the active site to the tracking site. This could be to allow site maintenance or perhaps as a test of the contingency procedures.

1. The decision is made to perform a takeover. The IMS /RTAKEOVER command is issued on IMSA. IMSB is stopped by a shutdown /CHECKPOINT command or a /RTAKEOVER command.

   If the tracking site needs to recover the IMS message queues, a /CHECKPOINT DUMPQ command should be issued on one system (say, IMSA) and a /RTAKEOVER DUMPQ command on the other (IMSB).

   When DBRC accepts the /RTAKEOVER command, no new subsystem is allowed to sign on to RECON/AB, and the status of RECON/AB is changed to tracking.

2. IMST receives notification of the remote takeover in the transmitted log data (SLDS/A).
3. All of the log data must be received by IMST. If any gaps exists, the ILS must send them before takeover can complete.
4. Once all log data has been received, IMST closes down.
5. An emergency restart of IMSA and IMSB is performed at the remote site.
6. The telecommunication lines and terminals are switched to the new active subsystems, and the terminals are then able to reconnect to IMS.

Unplanned Takeover

This process can only be initialized on the tracking site by IMST. Assume the location housing the IMSA system suffers a catastrophic failure:

1. The command /RTAKEOVER UNPLAN is issued on IMST.
2. If possible, the user must try to terminate IMSB.
3. The log router tries to receive all the log data it needs. In the best case, only the last data from IMSA is lost. The database trackers apply all possible updates, and then IMST shuts down.
4. An emergency restart of IMSA and IMSB is performed at the remote site. To rebuild the message queues, use /ERE BUILDQ. Otherwise, cold start the queues using /ERE COLDCOMM or COLDSYS.
5. The database recovery can then be performed if needed.

5.1.11 Implementing Remote Site Recovery

Recovery Plan

Before RSR can be successfully implemented, many other tasks must be performed:

- The RSR features must be installed on the two sites.
- New procedures for database recovery, system recovery, and network switching must be carefully prepared and tested on the tracking site.
- Some important choices must be made regarding database management:
  - Frequency of Image Copy at the active site, and procedures to transfer copies to the tracking site.

  - Frequency of Image Copy at the tracking site (for local recovery at the remote site)
  - In an RLT environment, the frequency of recovery at the remote site: periodic recovery (for example, nightly) or only if a takeover occurs.
  - Database reorganization management, to ensure synchronization between the remote and primary copies.
- An MVS and IMS system must be provided at the tracking site. The remote environment must be maintained with any changes (exit routines, procedures, EXECs) made to the active system.
- The network configuration must give enough flexibility to switch the connections from one site to the other and must support the new workload added by the RSR LU6.2 conversations.

Each installation must very carefully plan and test the remote recovery procedures. It must be remembered that an IMS service takeover, as well as requiring software and hardware solutions, also involves people and supplies.

After a takeover has been performed, and assuming the original site can be "repaired," the customer will probably want to return the IMS service to the original site. This requires a scheduled takeover and must be an equally well-documented and tested procedure.

Remote Site Recovery Setup

Two copies of the RSR feature are needed for a typical send and receive configuration. One RSR facility is needed for each MVS system.

The same release levels of IMS are required for the send and receive pair. Normally, the MVS and IMS systems should be similar enough that the output of stage 2 of IMS system definition can be copied from one subsystem to the other.

IMS Sysgen Considerations: To use the RSR feature, it must be requested at IMS system definition time, using the RSRFEAT parameter on the IMSCTRL macro. Valid values are NO|RLT|DLT. A global service group (GSG) name and a TMS name must be provided in the sysgen, in the IMS startup procedure, or in the startup PROCLIB member as well as in the DLI batch procedures. The APPLID parameter of the COMM macro now contains three names, one for the IMS active subsystem, one for the XRF subsystem, and one for the IMS RSR tracker subsystem.

The databases to be tracked must be included in the system definition (DATABASE macro) of the IMS tracking subsystem. This applies even if they are only used by batch jobs.

Refer to *IMS/ESA V5 Installation Vol. 2: System Definition and Tuning, SC26-8023* for more information. The VTAM definitions for the TMS are also described in that manual.

IMS Startup Parameters: The startup parameters for the IMS active subsystem and for the IMS RSR tracking subsystem are different. Some of the differences are mentioned here, but for complete information refer to *IMS/ESA V5 Installation Vol. 2: System Definition and Tuning, SC26-8023*.

In the IMS startup procedure, the TRACK parameter specifies the level of recovery tracking. TRACK=NO is used for the IMS active subsystems.

TRACK=RLT|DLT is used as the startup parameter for an IMS tracking subsystem.

The USERVAR parameter is required for RSR-capable active systems. It contains a user name used in the LU6.2 conversations.

IMS DFSRSRxx PROCLIB Member: This member contains the RSR options used by the online active and tracking subsystems in an RSR complex. The suffix is specified with the RSRMBR parameter in the IMS startup procedure. The two (or more) members used in the RSR complex do not need to be identical; but GSGNAME must be the same.

RSR and BLDS

In a BLDS environment, all IMS subsystems sharing databases use the same tracker.

The log router merges the logs for DLT. The change accumulate utility is used, as before, in the case of RLT.

Only one ILS is needed. It must have access to the shared RECONs and to the archived logs. It is associated with one of the TMS instances and fills the gaps for the whole group of sharers.

5.1.12 Migration

The use of the RSR feature is optional. It is compatible with all existing IMS facilities. Migration considerations apply to IMS 5.1 migration rather than to an RSR implementation, but there are implications for fallback:

- DBRC migration

  When the databases or areas are registered in DBRC for tracking, only IMS 5.1 systems should update them. Log data created by an older version of IMS would not be sent to the tracking site! This includes DLI Batch processing.

- Terminal considerations

  You should review terminal support for SLUP, FINANCE, and MSC-connected devices to ensure that terminal actions in an RSR environment perform as desired.

- MSDBs must be converted to DEDB, probably with the VSO option, to participate in remote recovery. (MSDB log records can be sent, and handled, by user procedures outside the scope of IMS RSR).

- GSAM files are not supported.

5.2 Enhanced Timestamp Recovery with DBRC

The use of timestamp recovery must be carefully controlled and considered because there is always the possibility of the destruction of a database if an unknown change has somehow occurred. The rules enforced by DBRC are designed to minimize the chance of such an occurrence. These rules have been relaxed as much as possible without compromising system integrity.

5.2.1 Overview

Customers typically use timestamp recovery to recover to a point-in-time prior to an application logic error or operational error (for example, using wrong input file in batch). The IMS Recovery Utility and DBRC impose certain rules concerning when timestamp recovery is possible. As a consequence, operating procedures must ensure that these rules are not compromised. (For example, attempting to minimize the number of log switches, while beneficial to the logging aspects of the operational environment, can create difficulties for timestamp recovery.)

In IMS 5.1, the rules have been relaxed, giving more flexibility and providing the potential for simpler operating procedures.

5.2.2 Description

The old and new rules for timestamp recovery are described below.

Timestamp Recovery before IMS 5.1

Figure 22 shows the DBRC rules governing the timestamp that can be used for timestamp recovery for IMS releases earlier than IMS 5.1.

1. No subsystem has the database open for update (that is, no open ALLOC record in DBRC).
2. The input logs to recovery contain no updates later than the timestamp.
3. There is no perceived need to merge input logs.

Figure 22. Valid Times for Timestamp Recovery before IMS 5.1

DBRC must check for a **merge needed** situation whenever the time period between the beginning and end of one input log data set overlaps that for another input log data set. Log records obviously have to be processed by the Recovery Utility in strict chronological order. DBRC selects the necessary log volumes (SLDS) and *orders them in Stop Time (data set end time) sequence*. If this would (*or might*) result in log records being presented in the wrong sequence, DBRC will insist on the logs being merged with the Change Accumulate Utility before recovery. However, change accumulate can discard updates when there are later updates to the same data, so timestamp recovery then becomes impossible.

Figure 23 on page 101 shows a situation in which an online system was updating a database until time T1. It was then taken offline but *no log switch was performed*. Then at T2, a batch update was begun that finished at T3. The database was brought back online at T4. The online log switch occurred at time T5.

Rule 1 limits the possible recovery times to the ranges T1-T2 or T3-T4. But recovering to any time in these two ranges will not comply with rules 2 and 3.

In other words, in IMS 4.1 and previous releases, timestamp recovery was not possible in this scenario.

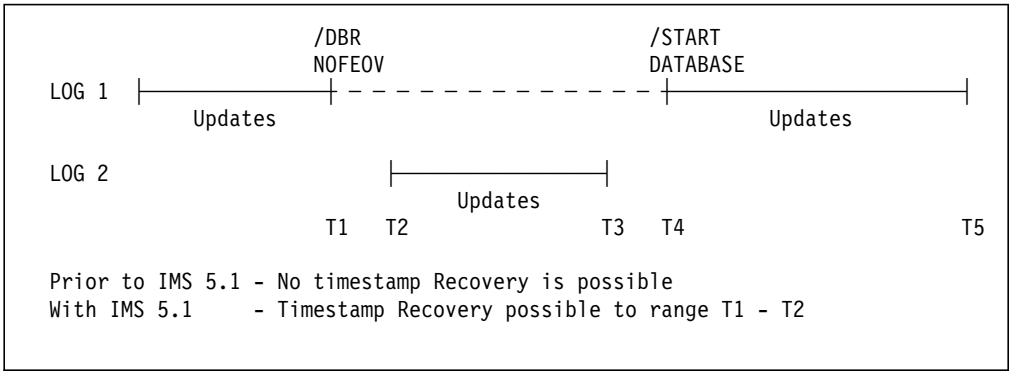


Figure 23. Timestamp Recovery Example 1

Figure 24 shows a similar situation except that the database is not brought online again after the batch update.

In this case, rule 1 indicates that recovery times in the ranges T1-T2 and T3-T5 are acceptable. Rule 2 does not prohibit either time range. However, DBRC sees the two logs as overlapping and hence needing to be merged[7]. However, it is only for recovery times later than T2 that this applies—prior to this, only one log is needed for input to the recovery utility.

Consequently DBRC will allow timestamp recovery to any time in the range T1-T2 but not after (and hence *not* T3-T5).

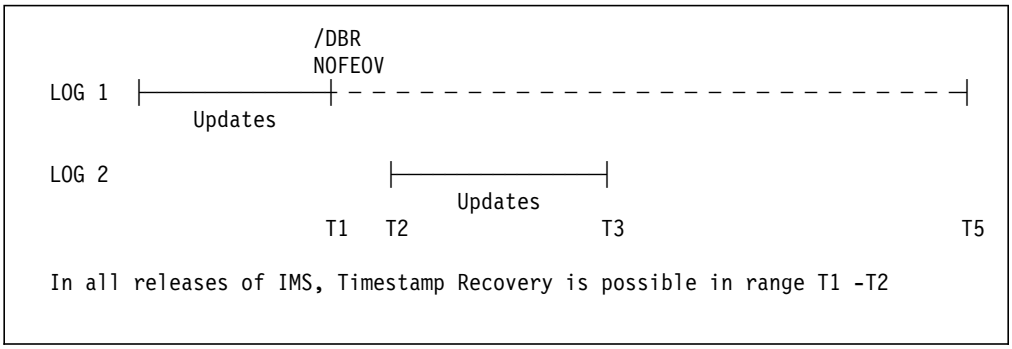


Figure 24. Timestamp Recovery Example 2

This illustrates a possible inconsistency in previous releases of IMS. If there is no log switch before the database is brought back online, timestamp recovery is not possible. If there is a log switch beforehand, timestamp recovery is possible.

[7] This is a consequence of /DBR NOFEOV rather than FEOV.

Timestamp Recovery in IMS 5.1

Only two rules are imposed on specifying a valid timestamp in IMS 5.1 (see Figure 25).

1. No subsystem has the database open for update (that is, no open ALLOC record in DBRC).
2. There is no perceived need to merge input logs.

Figure 25. Valid Times for Timestamp Recovery in IMS 5.1

Referring again to Figure 23 on page 101, rule 1 allows any time stamp in the ranges T1-T2 or T3-T4. In the range T1-T2, there has only been one log (LOG 1), and so there is no requirement to merge logs 1 and 2. Consequently, IMS 5.1 will allow timestamp recovery in the period T1 to T2.

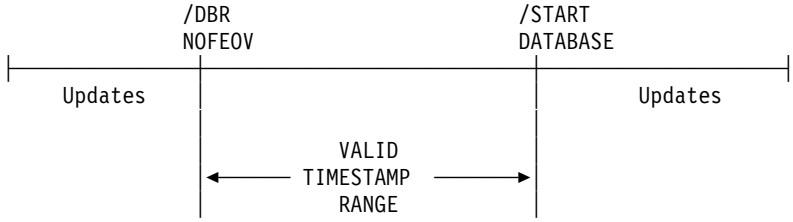
However, DBRC would not allow recovery to times in the range T3-T4 because two input logs are selected that overlap and "require merging."

Referring again to Figure 24 on page 101, the changes in IMS 5.1 make no difference. Period T1-T2 is still valid. Period T3-T5 is still invalid because of the perceived need to merge logs 1 and 2.

This illustrates that the inconsistency in previous releases has been removed in IMS 5.1. The ability to timestamp recover a database does not depend on a log switch prior to the database being brought back online.

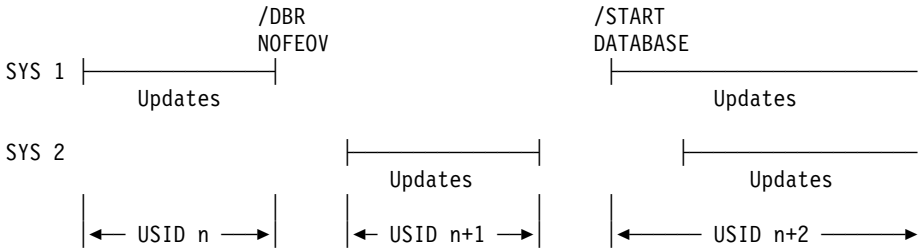
Update Set Identifiers

The rule that no longer applies for determining a valid timestamp is, "The input logs to recovery must contain no updates later than the timestamp." In other words, IMS 5.1 can use a timestamp that is inbetween two sets of log records on the same log data set, as long as the database was not open for update at that time.



Each time period during which a database or area is open for update on one or more IMS subsystems is given a unique numeric identifier, the USID. The USID relates to a single Full Function database or DEDB area. It is set each time a Full Function database is first updated after being opened or when a Fast Path Area is opened for update (that is, at DBRC ALLOC processing). The rules DBRC uses for setting its value are:

- If no other subsystem has an ALLOC record for the database or area, increment the current USID and return it to the new subsystem.
- If an ALLOC record already exists for the database or area, the current USID is returned to the new subsystem.



The USID is:

- Stored in the RECON in the database record (or area record)
- Included in each database update log record
- Understood by the Recovery Utility.

When a timestamp recovery is run under the control of DBRC, DBRC passes the appropriate USIDs to the Recovery Utility at execution time.

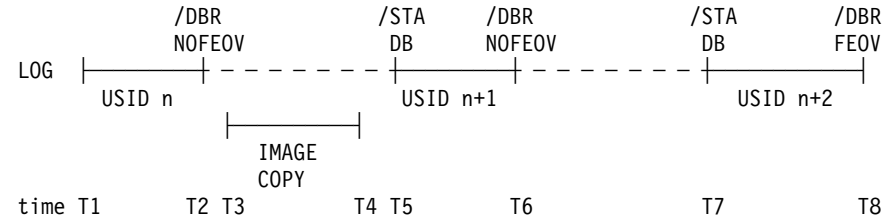
The Recovery Utility ignores log records with a USID outside the specified range. This is a significant enhancement over previous releases, where the Recovery Utility always applied every log record, with a timestamp later than the Image Copy, to the database (every complete UOR in the case of an area).

The Recovery Utility reports the time range of any unused log records:

DFS1941I STOPPED PROCESSING LOG WHEN TIME=yydddhhmmsst ENCOUNTERED

DFS1940I RECORDS SKIPPED FOR DBD=xxxx DDN=yyyy FROM TIME=yydddhhmmsst TO TIME=yydddmmsst COUNT=cccc

The DFS1040I message will be returned if the Image Copy has a time later than one or more USIDs on the log.



A timestamp recovery to time T6 would use the Image Copy and log shown and would produce:

- DFS1940I to indicate USID n had been skipped
- DFS1941I to indicate USID n+2 had been encountered.

Note that this is another example of a timestamp recovery that was previously impossible.

Note: The USID looks similar to the data sharing sequence number (DSSN). However, for Full Function databases, the DSSN relates to a data set whereas the USID relates to a database. For a DEDB, both the USID and the DSSN relate to an area, they are both incremented at the same time, but their values will differ by 1.

5.2.3 Migration

No user effort is required to implement the new timestamp recovery. The only point to be aware of is that the new timestamp recovery utility will only exploit the USID facility when no logs from a previous release are included.

There is no change to the Recovery Utility JCL or SYSIN data as a result of this enhancement.

5.2.4 Value

The USID facility and enhanced timestamp recovery are essential in an RSR environment. However, the new flexibility is available in any environment.

Timestamp recovery is usually necessary after some form of user error (for example, wrong input file, wrong version of program, two programs in wrong order), and as usual, there is often pressure to repair the system as quickly as possible. By enabling a wider range of timestamps, IMS 5.1 provides the potential for recovering to a later timestamp than had previously been possible. This in turn would reduce the number of application reruns and in some cases could reduce or eliminate transactions being reentered by end users.

5.3 Enhanced Message Error Handling

This user exit is applicable for message networking environments and should be viewed as mandatory for MSC systems requiring high link availability. It is highly recommended in all other environments.

The changes in IMS 5.1 build upon the original IMS 4.1 implementation and consolidate a variety of message traffic error conditions into a single point of control. This simplifies operational complexity and allows for more system managed function while allowing the right action to be taken based on the customer environment.

5.3.1 Overview

The MCEE (DFSCMUX0) was introduced in IMS 4.1 to handle /DEQ commands and to minimize the effects of various MSC error situations, thereby increasing the overall link availability. The message “suffering the error” is passed to MCEE, which decides what to do with it.

In IMS 5.1, the MSC functions of the exit have been enhanced and the exit has become more general purpose, supporting some APPC failures as well as MSC errors.

5.3.2 Description

MCEE Exit for MSC

IMS 4.1 Function: The MCEE is invoked at various times:

- /RSTART LINK
- /PSTOP LINK
- Send error
- Receive error

- /DEQ MSNAME
- VTAM lost session (error situation or by command).

For MSC messages, the exit can:

- Discard the message
- Reroute to a local LTERM
- Reroute to a local transaction.

A major result of using the MCEE is that errors other than MSC physical link failures do not prevent subsequent use of the MSC link.

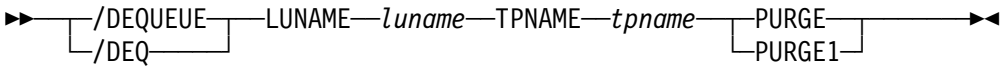
IMS 5.1 Enhancements for MSC: For MSC messages, the exit can:

- Discard the message
- Reroute to a local *or remote* LTERM[8]
- Reroute to a local *or remote* transaction
- *Reroute to an LU 6.2 destination (LUname plus TPName)*.

Therefore, there is now complete flexibility in the rerouting options. However, it is recommended that MSC rerouting always be done to a local non-conversational transaction that has been specially written to handle the various types of errors. It would be particularly unhelpful if a link problem caused the MCEE to reroute the message to another remote destination and subsequently experienced the same error!

MCEE for /DEQ Command

The MCEE in IMS 4.1 processed /DEQ LTERM, /DEQ NODE, and /DEQ MSNAME. It could suppress the command, reroute the message being dequeued, or allow the command to process. In IMS 5.1 it is also called when a new APPC-related dequeue option is used, namely:



The reroute options are the same as for MSC.

Change of Default Actions: In IMS 4.1, the default action for /DEQ MSNAME when no exit was coded was to suppress the dequeue command. This was for reasons of compatibility with previous releases. In IMS 5.1, the decision has been made to allow all /DEQ commands to be processed when there is no exit. To suppress any /DEQ command in IMS 5.1 requires the use of the MCEE.

MCEE for APPC/IMS

In IMS 5.1, the MCEE (DFSCMUX0) has been extended to include the function of the LU 6.2 Destination Exit Routine (DFSLULU0) that was introduced in APPC/IMS. *The DFSLULU0 exit is no longer available.* It was called when a synchronous APPC conversation SEND was issued by IMS and the LU 6.2 session failed. This represents a message integrity exposure since that APPC conversation cannot be restarted to receive the reply. So DFSLULU0 was

[8] The LTERM must be either a static LTERM or an already created ETO LTERM. IMS does not dynamically build a user structure in this case.

available to ensure that the message was delivered to an appropriate alternative destination (or the transaction backed out). This APPC exit function has been moved to the MCEE in IMS 5.1 due to the similarity with the MSC rerouting function.

The MCEE is called when an APPC SEND, issued by IMS, is rejected with a *deallocate with send error*.

The function of the exit in the APPC case is just as for MSC, namely, discard the message, reroute to any local or remote destination, or send it through an asynchronous APPC conversation to an LU6.2 destination.

5.3.3 Migration Considerations

IMS 4.1 supplied defaults for both DFSCMUX0 and DFSLULU0. IMS 5.1 supplies a default for DFSCMUX0, which incorporates the default actions of both exits from IMS 4.1.

The only change to the defaults is in the handling of /DEQ commands, as noted in “Change of Default Actions” on page 105.

If either exit was modified before use with IMS 4.1, it will be necessary to provide a new DFSCMUX0 in IMS 5.1.

5.3.4 Value

The new MCEE provides a single, flexible focal point for managing link and APPC session failures. It provides benefits in minimizing the impact of failures and in addressing the message integrity exposure in such cases.

5.4 MSC Forced Close of Link

The problem of a hung MSC link is particularly serious because it can affect multiple IMS systems and usually requires an IMS restart to cure. In the worst case an IMS cold start may be required for all IMS systems in the MSC network. This scenario should never occur in IMS 5.1 (some combination of the /DEQ command, MCEE, and the MSC force close option described here should be effective in resolving every MSC link problem).

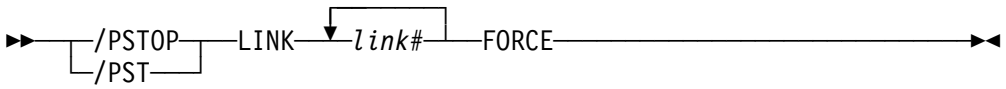
5.4.1 Overview

From time to time, many MSC users have had problems associated with VTAM errors that cause VTAM and IMS to have a different perception of a link′s status. In these cases, the link is unusable without an IMS restart. IMS 5.1 removes the need to restart IMS.

5.4.2 Description

In certain error situations, due to a lost VTAM event, an MSC link becomes unusable. With previous releases of IMS, no IMS command (/CLSDST, /IDLE, /PSTOP) or VTAM command (VARY NET,INACT,FORCE) would rectify the situation. The only solution in IMS 4.1 was to restart the IMS system.

A new option, FORCE, on the IMS /PSTOP LINK command has been introduced in IMS 5.1 to clean up the link control blocks and so enable the link to be restarted without an IMS restart:



When this command is entered, IMS checks that the link is a VTAM link in a PSTOP Pending state (PSTOPPED NOTIDLE) and that the session is terminated. If so, the control blocks are cleaned up and the link status is set to PSTOPPED IDLE.

5.4.3 Implementing /PSTOP LINK FORCE

Operating procedures should be updated to include the new command usage. If the command is ineffective, it is probably because the session has not been terminated. The operator must first identify the session and terminate it, using the appropriate VTAM commands:

```
D NET,SESSION,LU1=applid1,LU2=applid2,SCOPE=ALL,LIST=ALL
V NET,SESSION,SID=sess-id,NOTIFY=YES,SCOPE=ALL,TYPE=FORCE
```

It may then be necessary to reissue the /PSTOP LINK FORCE command.

5.4.4 Migration Considerations

FORCE is not a valid parameter on a /PSTOP command in previous releases of IMS.

5.4.5 Value

Many customers will have experienced the odd occasion when IMS has had to be closed down and restarted to clean up an MSC link. It may even be that this had to be done during the online day, to allow essential MSC traffic to be resumed. The new facility in IMS 5.1 enables the IMS service to continue uninterrupted during the resolution of link problems.

5.5 Dynamic Update of IMS Type 2 SVC

5.5.1 Overview

Each release of IMS on an MVS system requires a unique type 2 SVC. The installation of this IMS SVC has previously required an MVS IPL. IMS 5.1 includes a new utility to install an IMS SVC without an MVS IPL.

5.5.2 Description

The SVC Utility (DFSUSVC0) allows the user to add or replace an IMS type 2 SVC dynamically. The IMS SVC number is supplied by the customer. (The actual process is that the customer puts the SVC number into the IMS sysgen, the sysgen creates a RESLIB, and the RESLIB is input to the SVC utility.)

Before running the utility with the new RESLIB, the old RESLIB containing the old SVC should be secured in case of fallback.

All IMS activity using the SVC to be modified *must* be stopped before running the new utility. DFSUSVC0 first performs a series of comprehensive checks of the environment and issues WTOs or WTORs for the following conditions:

- WTOR - There is no existing SVC for this SVC number.
- WTOR - The existing SVC is not an IMS SVC (may be an MVS dummy SVC).
- WTOR - The existing IMS SVC is at a different release level.
- WTO - One or more IMS systems are currently using this SVC (utility will automatically abort).

When all of the checks are satisfied (customer replies yes to any WTORs), DFSUSVC0 loads the SVC module from IMS.RESLIB into MVS global storage and updates the MVS SVC vector table with the new SVC address.

The SVC utility does not remove the need to add the IMS SVC to the MVS nucleus. Every MVS IPL will regress the customer back to an old IMS SVC and requires using the utility to reinstall a new IMS SVC.

This utility should be used with care—it is essentially an authorized program modifying MVS tables.

5.5.3 Value

In general, IMS SVCs are very stable. However, when installing a new release of IMS, it is usual to have both the old and the new SVCs coexisting. In the past, adding the new SVC has required MVS to be shut down and re-IPLed. For some customers, even such a planned shutdown of MVS is unsatisfactory. The IMS 5.1 utility enables a new type 2 SVC to be added to MVS without interrupting the general MVS service.

Chapter 6. IMS 5.1 Distributed Processing and Open Access

This chapter describes the IMS 5.1 enhancements that relate to distributed processing and open access. These are the new Open Transaction Manager Access facility, POSIX compliance, MSC support for transactions originating at APPC devices, and other MSC enhancements.

6.1 Open Transaction Manager Access

Open Transaction Manager Access (OTMA) is a base component of IMS 5.1 that provides an access path and an interface specification for sending and receiving transactions and data from IMS.

Traditionally, IMS transactions have been entered at a terminal on a Systems Network Architecture (SNA) network, transmitted to an MVS application (VTAM), and then passed on to IMS by VTAM.

Now other types of networks (non-SNA) need to be supported, the most notable being TCP/IP. The MVS based product supporting the TCP/IP network is TCP/IP for MVS. Clearly then there is a requirement for IMS to communicate with TCP/IP for MVS.

Further, three emerging standards for client/server processing (conversational, remote procedure call, and messaging and queueing) are supported on the MVS platform, as follows:

Conversational	APPC/MVS
DCE/RPC	MVS OpenEdition + Application Server/IMS
Messaging and Queueing	MQSeries for MVS

It is very important for IMS to support all three of these programming models.

IMS 4.1 fully supports the conversational model using the LU6.2 protocol through VTAM and APPC/MVS. In IMS 5.1, OTMA provides the facility for IMS to communicate very efficiently with MVS applications other than VTAM, such as TCP/IP for MVS, Application Server/IMS, and MQSeries for MVS.

Refer to 8.5, “The Open IMS Server” on page 152 for more information about this open world.

OTMA communicates with MVS applications using MVS′s high-performance XCF function. OTMA and the MVS applications compose an *XCF group*, where OTMA and the applications are *group members*. In a Parallel Sysplex environment, different members can be on different MVS images.

OTMA, through its protocol layer above the XCF API, provides to the other group members an interprocess communication mechanism for accessing IMS applications (see Figure 26 on page 110).

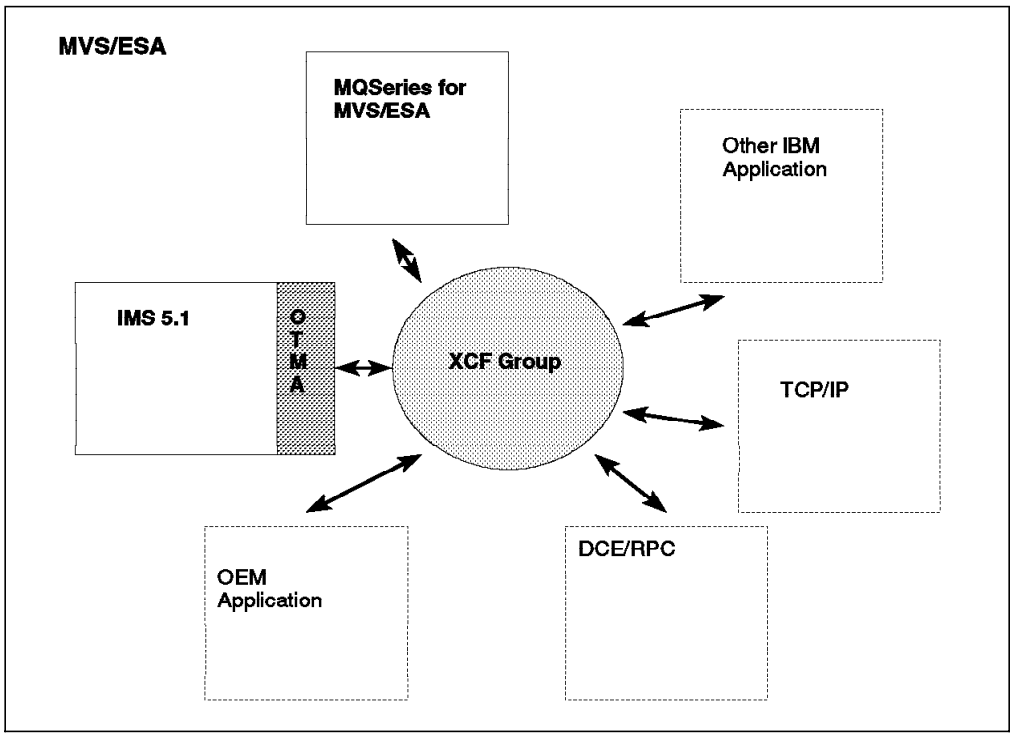


Figure 26. XCF Utilization with OTMA

Tpipes Concept

In IMS, the logical terminal or LTERM construct is used to provide a queue where an output message can wait before being transmitted. IMS knows how to associate this LTERM with the physical node that has to receive the message.

With the OTMA interface, the MVS client application program specifies a *Tpipe* name to indicate the queue where it wants IMS to put the output messages. The client can specify the same Tpipe for many input messages. A client token (or correlation token) can be included with the message, to allow the client to correlate each reply from IMS with the original input message.

Different OTMA clients can use the same Tpipe name as an anchor, because the structure that is created in IMS to keep the messages relies on two names, the client name and the Tpipe name. The client name is actually the member's name when the client joins the XCF group. It is provided directly to IMS by the XCF interface.

With this implementation, the client has control over the output of its transactions, and IMS does not know anything about the transaction end user. The client also dynamically chooses the parameters that control the flow of the messages on a Tpipe.

Tpipes provide a full-duplex mode of communication, but the client can implement a half-duplex-like processing (like an LTERM) by:

- Specifying a token to correlate input and output messages
- Forcing IMS to send an ACK when it receives the input message
- Designating a Tpipe as *synchronized*.

In OTMA terminology, “synchronized” implies “recoverable,” and message sequence numbers are implemented to allow the client application to request resynchronization processing. But a Tpipe does not need to be synchronized to allow messages to be recoverable.

Services

OTMA provides transaction-specific services to the client such as the ability to:

- Query the server to know which transactions are supported and the server′s transaction attributes (architected /DIS TRAN command)
- Decide how transaction requests and responses are to be processed in the server
- Include user data with the transaction and allow that data to stay with all of the transaction-generated messages
- Specify a client token with each input transaction. IMS returns the token with the reply, allowing the MVS client to associate that reply with the correct remote client
- Correlate the subsequent transactions within an IMS conversation by using the IMS-generated server token to identify the conversation instance.

OTMA multiplexes all client transactions over a single transport connection. OTMA assumes that both the clients and the IMS server create a single XCF connection when joining the XCF group.

Supported Functions

The majority of IMS message processing options can be implemented with OTMA. The supported functions are:

- Nonresponse mode and response mode transactions
- IMS conversational processing
- Fast Path transactions
- MSC processing
- IMS commands (not all)
- XRF class 3 support (active and alternate in the same complex)
- IMS Message Requeuer V2 (5655-038).

OTMA does not support MFS (although MODname is supported on both input and output messages) and some IMS commands.

Message Structure

OTMA messages have the following syntax (see Figure 27 on page 112):

- Message Control Information (MCI) section

  This section contains many important fields, such as Tpipe name, type of message, sequence numbers for segment ordering and message recovery, the processing flag, and the multisegment chaining indicator (FIC, MIC, LIC, OIC).

- State Data section

  This section contains fields used by either the client (on input) or IMS (on output). It includes Destination Override, Map name (MODname), Sync_Level, Commit Mode, Tokens, and the Server State.

- Security section

  This section is mandatory on every transaction and can be present on OTMA command messages.

- User Data section (managed by the client)
- Application Data section

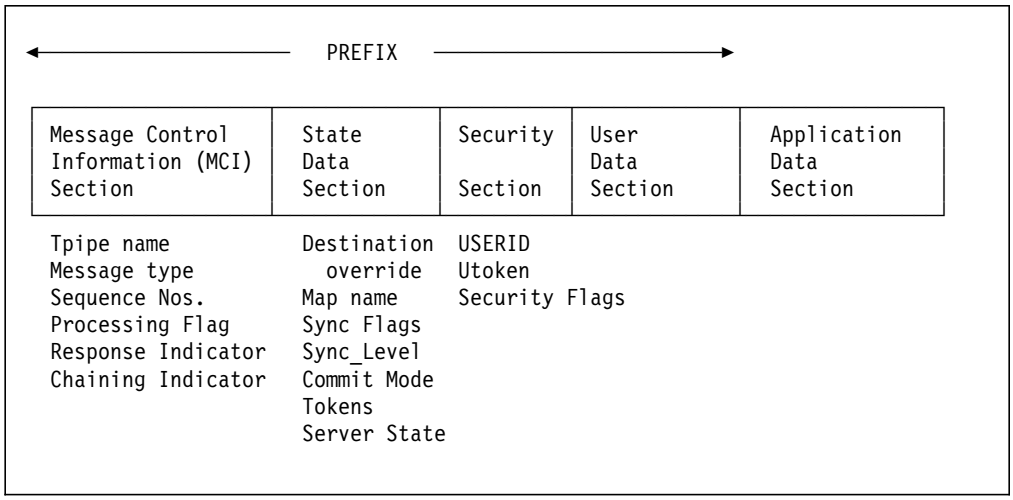


Figure 27. OTMA Message Structure

The key parameters that affect the nature of the transaction processing are:

- **Commit Mode** (set in Synchronization Flags)

0	Traditional IMS flow. Reply is enqueued and sent after syncpoint.
1	Like APPC synchronous flow. The reply must be successfully sent before syncpoint processing begins.

- **Sync_Level**

NONE	Replies are sent by IMS to client application without requesting an acknowledgment.
CONFIRM	Client is asked to acknowledge all IMS output.

- **Processing Flag**

  X'40' is used to specify whether a Tpipe should be synchronized.

- **Client Token (Correlator Token)**

  Used to match an input message and its reply.

- **Server Token**

  Used to associate the multiple phases of an IMS conversation.

Not all combinations of parameter values are valid. See Table 8 on page 117 for more information.

Message Flow: OTMA Client to IMS Server

Figure 28 illustrates the OTMA client to IMS server flow and the classical LU2 flow.

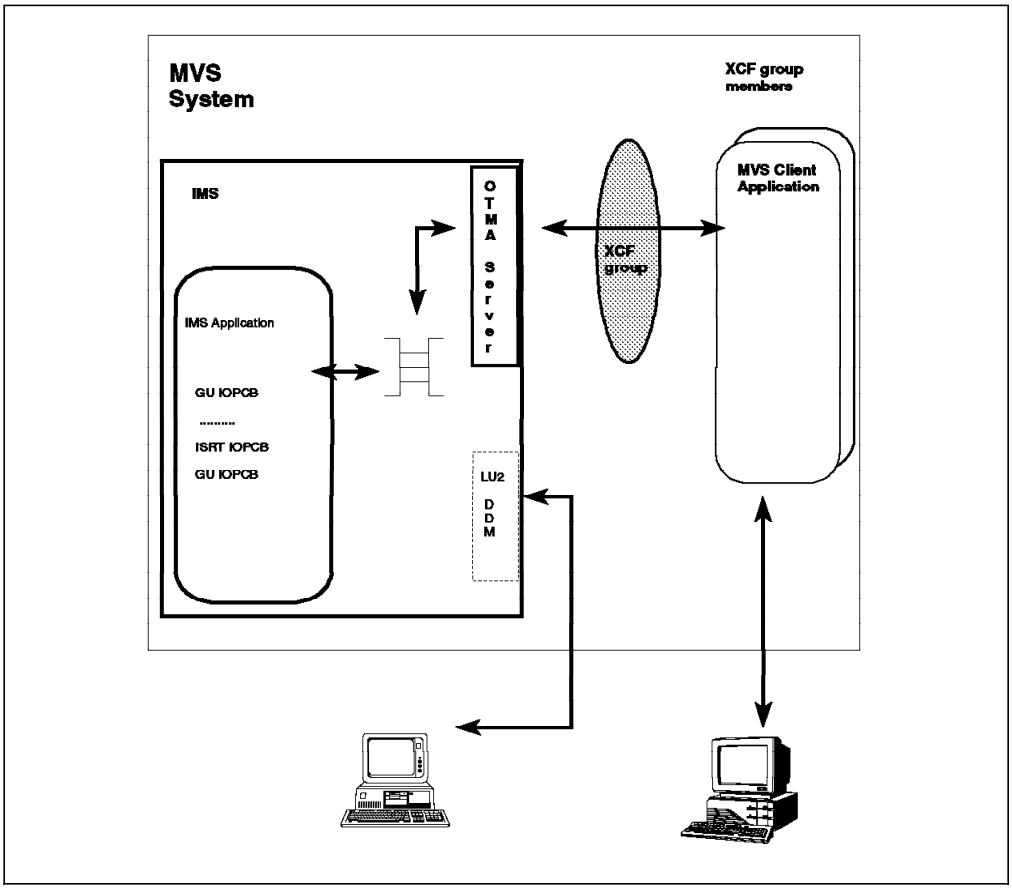


Figure 28. Message Flow: OTMA Client to IMS Server

The general message flow is as follows:

1. The OTMA client submits an IMS transaction or an IMS command through XCF. The IMS transaction code or command is specified in the Application Data section of the input message. A message prefix is always attached to the input message. It contains the Tpipe name, the commit mode, client and server tokens, and many other flags (see Figure 27 on page 112).
2. The input message is enqueued on the SMB for a transaction or is processed by IMS for a command.
3. The LTERM field from the IOPCB contains the **Tpipe name** specified in the Message Control Information section of the input message, or the LTERM field is overridden by the **Destination override** specified in the State Data section of the input message.

   The client can also provide a MODname in the **Map name** specified in the State Data section of the input message. It will be placed in the IOPCB. Similarly, this prefix field will be updated in a reply if the IMS application specifies a new MODname in the ISRT call.
4. The IMS application cannot see the OTMA prefix when it issues the GU call.
5. For IMS commands, output is sent back synchronously with a few exceptions.

For IMS transactions using commit mode 0, application output is enqueued on a dynamically created IMS Tpipe structure before being sent to the client.

For IMS transactions using commit mode 1, the output is sent synchronously by IMS and bypasses the message queues.

6. The sync_level determines whether the application requests an acknowledgment from the client when sending commit mode 1 output. (In commit mode 0, acknowledgments are always requested.)

Because XCF cannot guarantee sequential delivery of messages, IMS ensures that incoming segmented messages are ordered correctly.

Message Flow: IMS Client to OTMA Server

An IMS application program can send a message through an alternate PCB to an MVS application for routing to a terminal. The MVS client recognizes the message as an unsolicited output, because the message prefix only contains the Tpipe name. OTMA routing of alternate PCB output requires the use of the Pre-Routing Exit (DFSYPRX0), which will choose between OTMA processing and normal IMS processing for this output message. See “Pre-Routing Exit (DFSYPRX0)” on page 121 for more information.

The client application specifies the type of syncpoint processing it wants IMS to perform by setting the commit mode in the Synchronization Flags in the message prefix. If the option is incompatible with the type of Tpipe, it receives a negative acknowledgment (NACK).

Commit Mode 0: This mode causes IMS to commit output by inserting it to the IMS message queues *before sending it* to the client application. This flow is similar to an LU6.2 asynchronous conversation with APPC/IMS. The output message is then recoverable. It can be sent multiple times if acknowledgment is not sent by the MVS client.

In the case of a nonsynchronized Tpipe, the input message is recoverable or not depending on how the transaction is defined in IMS. In the case of a synchronized Tpipe, the input message is always recoverable.

Figure 29 on page 115 illustrates the typical commit mode 0 flow, which is also called the “commit then output” or “standard” flow.

With the standard flow, the Tpipe may be synchronized so that IMS maintains sequence numbers and then allows resynchronization. In addition, an acknowledgment is always requested by IMS and by the client. IMS ignores the sync_level (NONE or CONFIRM) in this case.

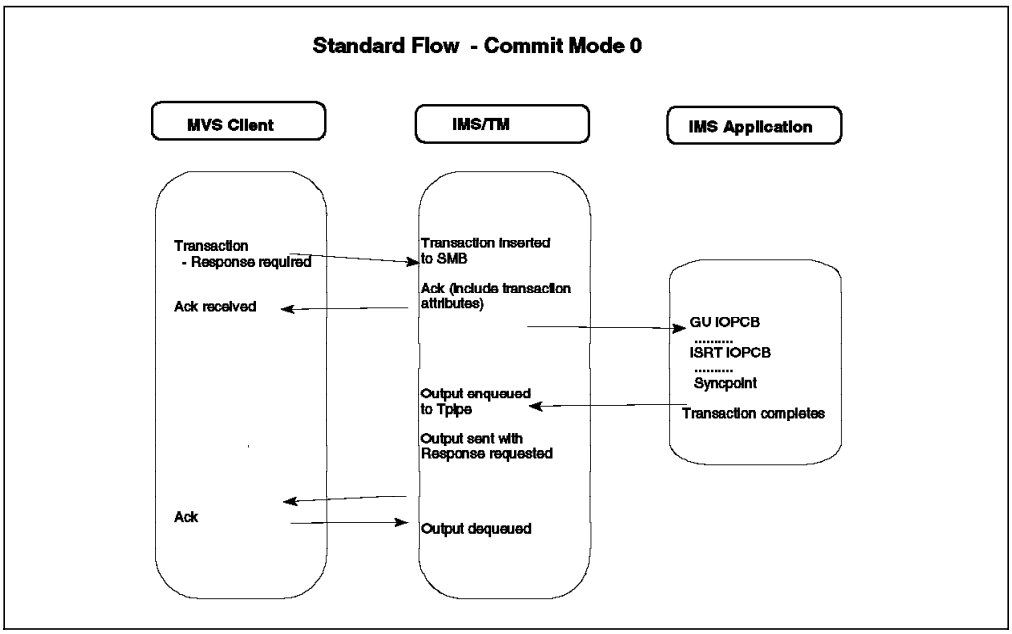


Figure 29. OTMA Commit Mode 0 Flow

Commit Mode 1: Figure 30 illustrates the commit mode 1 flow, which is also called the "output then commit" flow. Its main characteristic is the sending of IMS output before IMS process syncpoint.

With the output-then-commit flow, the Tpipe must not be specified as synchronized. **Force Response** is set (in the Synchronization Flags in the State Data section of the message prefix) to guarantee that the client receives some form of output data, even if the server transaction program ends without sending data to the client. DFS2082 is sent if the application terminates normally without sending output; DFS555 is sent if the region terminates abnormally. Depending on the **sync-level** (NONE or CONFIRM), an ACK may be requested.

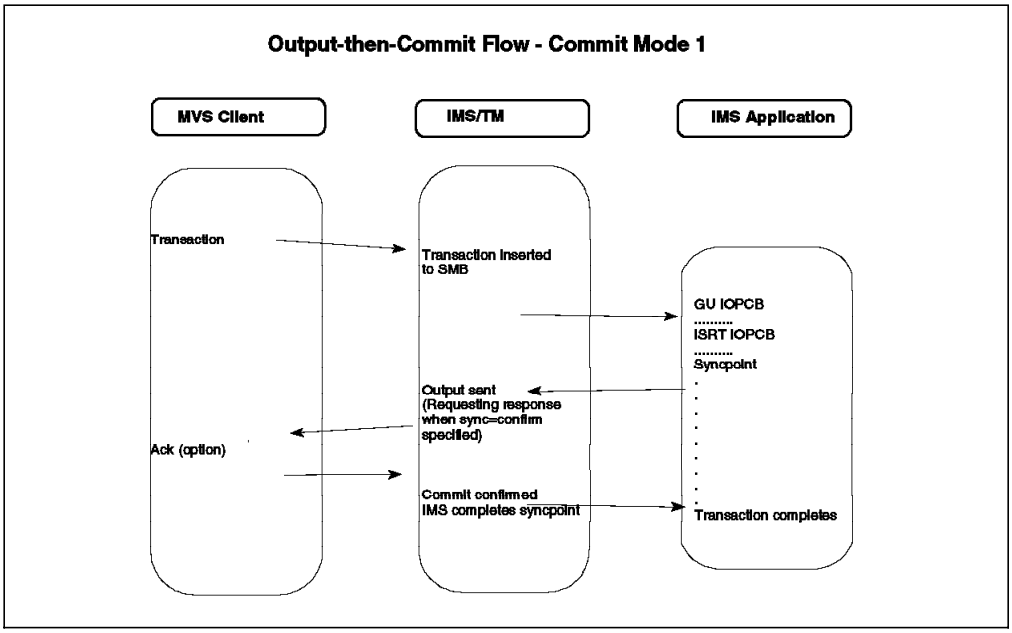


Figure 30. OTMA Commit Mode 1 Flow

OTMA commit mode 1 must be used for IMS conversational and Fast Path EMH transactions and is recommended for response mode transactions.

With commit mode 1, the transaction reply does not use the message queues. It is sent directly (synchronously) to the MVS client before syncpoint processing begins and IMS (and the application) wait for the acknowledgment. Region occupancy is an issue here. If the MVS client acknowledges the reply quickly, the application will sit idle in the region for only a short time, but message integrity will then be the responsibility of the MVS client. If the MVS client first waits for an acknowledgment from the remote client, higher integrity is achieved, but region occupancy could be seriously affected.

Commit mode 1 output is nonrecoverable. Message integrity relies on the client securing a reply before acknowledging it.

Tpipe Synchronization

To provide message recoverability with commit mode 0, a Tpipe can be defined by the MVS client application as *synchronized*. Synchronized Tpipes are thus used to implement STSN-like recovery, as IMS supports the SLUP/3600/ISC terminals.

To define a synchronized Tpipe, two parameters should be specified in the OTMA message:

- The **Processing Flag** set to “synchronized Tpipe” in the Message Control Information section of the message prefix.
- Commit Mode 0 in the **Synchronization Flag** in the State Data section of the message prefix.

A synchronized Tpipe has the following characteristics:

- Typically, each remote client would be allocated a different Tpipe between the MVS client and IMS.
- All output is sequenced and delivery is serialized through a single process. IMS processes all output messages in FIFO order. Subsequent messages are sent only after ACK from the client for the preceding one. A NACK stops all output messages on that Tpipe.
- While this output processing is taking place, the client could be sending input messages to IMS (full-duplex processing).
- To have a half-duplex-like communication, the MVS client should not submit another transaction until the previous one has been acknowledged or the MVS client has to control the sending of transactions by coordinating them with the receipt of the IMS output messages.

All messages sent over a synchronized Tpipe are expected to specify “synchronized Tpipe” in the prefix and are recoverable.

Note: On a nonsynchronized Tpipe, transactions can be submitted at any time, without having to wait for the acknowledgment of the previous transaction.

Commit Mode Summary

Table 8 summarizes the two commit mode processes.

Table 8. OTMA Commit Mode Summary

	Commit Mode 0	**Commit Mode 1**
Synchronized Tpipe	Optional	No
Sync_level	Ignored (uses CONFIRM)	NONE or CONFIRM
Response mode transactions (nonconversational, non-Fast Path)	IMS commits after enqueuing the output to the client Tpipe. The output is delivered later.	IMS sends output to client and then processes syncpoint.
Non-response-mode transactions	IMS does not need to send output to the client. If output available, processing is as above.	Client should specify **Force Response**. Processing is as above.
Conversational	Not supported	Must be specified
Fast Path	Not supported	Must be specified
Enqueue the input	Yes	Yes
Enqueue the output	Yes	No
Recoverability	Possible	No
Resynchronization	Possible	No

Typical MVS Clients

Two likely combinations of OTMA message characteristics are:

1. Commit Mode 0, with
   - Synchronized Tpipes
   - One Tpipe per remote client (terminal) and hence many Tpipes between MVS client and IMS
   - Recoverable output provided by IMS
2. Commit Mode 1, with
   - One Tpipe for the MVS client
   - Correlation token used to associate replies with remote clients
   - Output messages made recoverable by MVS and/or remote client.

Distributed Syncpoint Processing

As for APPC/IMS, OTMA requires the services of MVS to perform distributed syncpoint processing. Until this is provided by MVS/ESA, IMS OTMA cannot support a two phase commit process in distributed syncpoint.

Security

Some design considerations are relevant to security for OTMA-originated transactions.

- The resource class, OIMS, must be specified in the RACF definition. If some other security product is used, a similar definition must be implemented. For test purposes, the option of not having any security tool is available but is not recommended for a production environment.
- When a client joins the XCF group, a separate task control block (TCB) structure is created for it.

- The client applications must be authorized.
- Security options can be changed on a transaction basis.
- When the client first attempts to establish the connection with IMS, IMS performs security checking based on the information contained in the security section of the *client-bid* request.
- The client can indicate, on a transaction-by-transaction basis, that no security checking is to be done during transaction processing, and so minimize the security processing overhead. This is specified in the **Security flag** in the Security Data section of the OTMA message prefix.

  If this option is not chosen, userids must be RACF-verified by IMS.

  Note: There is no way to bypass security when the client first attempts to establish the connection with IMS (client-bid time).

For detailed rules, refer to the *OTMA User's Guide*.

Message Sequence

XCF does not guarantee that messages are delivered in the correct sequence. So the MVS client application must deal with this possibility by using its own XCF Message Exit routine. As well as for output messages, client transaction messages may not arrive in IMS in the same sequence in which they were issued by the client.

Resynchronization

Resynchronization is only performed for commit mode 0 transactions on synchronized Tpipes.

A synchronization mechanism must ensure that a client request has been executed only once or not at all. OTMA provides a mechanism to allow the client and the IMS server to reestablish message flow on a synchronized Tpipe after a client or server outage.

The resynchronization process does the following:

- Prevents data from being reprocessed, by using the **send sequence number** and the **recoverable sequence number** in the message control information section of the message prefix.
- In the case of a lost input message, it detects the fact that IMS did not receive the data sent by the client and asks for it to be resent.

  Because IMS always asks for an acknowledgment before dequeuing its messages, the client cannot ask for a message to be resent when it has already sent the ACK.
- Recognizes that resynchronization may not be possible.
- Allows the client to decide which actions are to be taken to resynchronize.

6.1.1 Implementing OTMA

Using OTMA in an IMS system is quite straightforward. The definition and generation requirements to activate this capability are minimal. The considerations described in the sections that follow should be evaluated for the proposed application usage of OTMA.

IMS System Generation

Sysgen specifications for the OTMA implementation are not necessary.

Startup Parameters

These are the new control region parameters supported with OTMA:

- GRNAME = Name of the XCF group

  IMS creates or joins this XCF group during IMS initialization or after issuing the /STA OTMA command.

- OTMA = YES|NO

  To enable the OTMA facility, OTMA=Y must be specified; if OTMA=N, the /STA OTMA command must be issued.

- The IMS XCF member name comes from the USERVAR field in the IMS startup parameters. If none is specified, the IMSID is used.

The XCF group name and the IMS member name must remain unchanged at an IMS emergency restart.

Refer to *IMS/ESA V5 Installation Vol. 2: System Definition and Tuning, SC26-8023* for more details about these startup parameters.

Transaction Definition

OTMA does not require any transaction definition changes, but note the following:

Message Recoverability: Message recoverability is applicable only for commit mode 0 transactions.

Any input messages on the message queue that are marked nonrecoverable are discarded during an IMS restart. Any input messages on the message queue that are marked recoverable are left on the message queue and are eligible for scheduling after IMS restart completes.

Transactions from OTMA clients are recoverable if the recoverable sequence number in the message control information section of the message prefix is not 0. IMS then forces the transaction to be recoverable even if the TRANSACT macro does not contain the INQUIRY=RECOVER parameter. Synchronized Tpipes provide a way of managing the recoverable sequence number.

Using an architected /DIS TRAN command specially provided for OTMA, the client can discover whether an IMS transaction is recoverable or not. It can then take the appropriate action.

Response Mode Transactions: For the transaction defined in the IMS system definition as response mode, it is recommended that the client use commit mode 1. Messages are nonrecoverable. When IMS sends the output, it turns the value of the **Server State** in the State Data section to X′40′ (response mode).

IMS Conversational Transactions: Any transaction defined in the IMS system definition as conversational must be called by the client in commit mode 1 and is nonrecoverable. When IMS sends back the output, it sets the value of the **Server State** in the State Data section to X′80′ (conversational state). This flag must be set on all client input during the conversation. On subsequent

conversational input, the client also must return the **Server Token** provided by IMS in the first output message of the conversation.

Fast Path Transactions: The client must call all Fast Path EMH transactions in commit mode 1. The transaction is rejected if any parameter of the request is set up improperly.

OTMA Descriptors

OTMA uses the descriptor concept introduced in IMS 4.1 for ETO and APPC/IMS. OTMA descriptors are optional; they are used to associate a Destination Resolution User Exit (DFSYDRU0 or any name) with each client name. If a descriptor is not specified for a client name, the default exit routine, DFSYDRU0, is used. See "Destination Resolution User Exit (DFSYDRU0 or any name)" on page 121 for more information.

IMS Commands

XCF group members can enter a subset of the IMS commands. The /DIS TRANSACTION command has a specially formatted response when an OTMA client issues it. All other IMS commands are processed normally, and output is returned to the client. With a few exceptions, IMS commands must be submitted using commit mode 1. For detailed rules, refer to the *OTMA User's Guide*.

The following commands can be helpful in an OTMA environment:

Command	Comment
/DEQ TMEMBER tmembername TPIPE Tpipename PURGE|PURGE1	Dequeue messages from a Tpipe structure
/DIS OTMA	One status line for each XCF group member
/DIS TMEMBER tmembername TPIPE Tpipename|ALL	Display the status of Tpipes
/DIS STATUS TMEMBER	Display the Tpipes that are stopped
/DIS SHUTDOWN STATUS	Display shutdown status for all OTMA clients that have not yet terminated
/DIS ACTIVE	OTMA information is displayed in IMS 5.1
/STO TMEMBER tmembername TPIPE Tpipename|ALL	Causes IMS to send a suspend input for Tpipename OTMA command to the client and prevents IMS from sending output to it.
/STA TMEMBER tmembername TPIPE Tpipename|ALL	Causes IMS to send a resume input for Tpipename OTMA command to the client and resume sending output to it.
/STO OTMA	Cause IMS to leave the XCF group
/STA OTMA	Cause IMS to join the XCF group, if OTMA=Y has been specified
/SEC OTMA	Control the RACF security level
/TRA SET ON	OTMA traces are now available
/DIS TRACE TMEMBER	Display the Tpipes being traced

Architected /DIS TRAN Command: When an OTMA client issues a /DIS TRAN command, a fixed format reply is generated consisting of a repeating group of transaction data fields. Each group contains the transaction name; type (for example, Fast Path Exclusive, MSC, normal), flags (for example, response mode, conversational, multi-segment); and status (for example, stopped).

Exit Routines

There are two OTMA exit routines. They are used to handle asynchronous output and unsolicited messages.

Pre-Routing Exit (DFSYPRX0): An IMS application can send output from non-OTMA-originated transactions to OTMA clients only when the Pre-Routing Exit (DFSYPRX0) is available.

After an ISRT to an alternate PCB, DFSYPRX0 is the first exit called in OTMA destination determination processing. Its purpose is to choose the domain, OTMA or normal IMS, in which the message will be processed. When OTMA processing is selected, the client name is determined as follows:

- If the original message was from an OTMA client, the same client name must be used for all output. No client name overrides are allowed in the case of an OTMA-originated transaction.
- For non-OTMA-client-originated transactions, by the Pre-Routing Exit (DFSYPRX0).

Then the Tpipe name is set by the Destination Resolution User Exit.

Destination Resolution User Exit (DFSYDRU0 or any name): Each client application has a Destination Resolution User Exit associated with it. For a client that is not yet connected to the XCF group, OTMA descriptors can be used to associate a Destination Resolution User Exit (DFSYDRU0 or any name) with a client name.

When a client connects to the XCF group (client-bid time), each XCF client defines a Destination Resolution User Exit (DFSYDRU0 or any name), which IMS loads before acknowledging the bid. This DRU name overrides the name specified in the OTMA descriptor.

When a message is inserted into an OTMA client destination, IMS must determine whether the queue exists. If the queue does not exist, IMS must create the Tpipe to which it queues the output message.

If the Pre-Routing Exit (DFSYPRX0) has chosen OTMA processing, the Destination Resolution User Exit (DFSYDRU0 or any name) routine is called before the final destination is determined. DFSYDRU0 can choose to route the output message to an OTMA destination or a standard IMS destination (LTERM, transaction, LU6.2 destination). If OTMA processing is still chosen, the final destination is set as follows:

- The Tpipe name is equal to the LTERM name of the alternate PCB.
- The client name is provided by Pre-Routing Exit (DFSYPRX0).

XCF Exits: Some XCF exits are available and their usage is detailed in the *OTMA User's Guide*. Here are the available exits:

- Group Exit: This exit is called after various XCF events.

- Message Exit: Arrival of a message is signaled to the destination member by the scheduling of its message exit.
- Status Exit: This optional exit can find the status of group members.

6.2 POSIX Compliance

The implementation of POSIX support within MVS is shown in Figure 4 on page 15. More information can be found in the following MVS documentation:

- *GC23-3010-01 Introducing OpenEdition MVS*
- *GC23-3011-01 OpenEdition MVS POSIX.1 Conformance Document*
- *GC23-3012-01 OpenEdition MVS POSIX.2 Conformance Document*
- *GC23-3021-00 OpenEdition MVS System Service*.

The IMS documentation contains nothing on POSIX services (there are no restrictions on using POSIX within IMS).

6.3 APPC/IMS MSC Support

The APPC/IMS support first shipped in IMS 4.1 has been enhanced to include the capability to use MSC. This capability may be particularly significant for migration from the LU 6.1 Adapter for LU 6.2 (which could use MSC, albeit without message integrity). The APPC/IMS component provides LU6.2 support that is superior in every capability to the Adapter, and users of the Adapter should migrate to APPC/IMS as quickly as possible.

MSC enables communication between IMS systems included in the same MSC network, on the same or different MVS images. It allows transactions entered on one IMS system to be processed on another IMS system, transparently to the terminal user. System definitions on each IMS system describe the routing for the messages. Message routing is automatic, unless you use a routing exit routine to alter a message′s destination. MSC switches message from one IMS message queue to another IMS message queue.

In IMS 4.1, APPC/IMS was introduced, but it was incompatible with MSC. Switches to remote destinations were not allowed when the originating message was from an LU6.2 device.

Now when using MSC links between IMS 5.1 systems, the restrictions on transactions originating from LU6.2 devices are removed for:

- Application programs using APPC/IMS implicit support (DL/1 calls)
- The LU6.2 message switch facility (DFSAPPC service).

Explicit mode transactions (CPI-C driven) cannot exploit MSC facilities.

The new Input Message Routing Exit (DFSNPRT0) functionally replaces the Terminal Routing Exit (DFSCMTR0); it is called for LU6.2 and OTMA input messages, as well as for standard messages. See 6.7, “Input Message Routing Exit (DFSNPRT0)” on page 132 for more information.

6.3.1 Terminology

The terminology used to define the different IMS systems in an MSC environment is:

- The terminal or LU6.2 remote program sends a message to the **originating** system, which is also called the **local** system.
- An **intermediate** system is one that routes messages between the local and remote systems.
- The **remote** system is the system where the remote transaction executes.

APPC/IMS MSC support allows several new types of multisystem operation:

- Remote transaction processing
- Program-to-program switch to a remote transaction program
- Message switch.

6.3.2 Remote Transaction Processing

An LU6.2 program can initialize a conversation with a transaction program name for an IMS remote transaction.

This remote transaction is included in the sysgen and therefore can only be a standard or modified application program, not a CPI-C driven application program. All IMS transaction types, except Fast Path, are supported (response or nonresponse mode, conversational or nonconversational).

The local IMS system provides the following services for the remote LU6.2 application program:

- Receives the inbound conversation allocation request.

  Accepts the LU6.2 conversation with the APPC application program.
- Calls the Input Message Routing Exit (DFSNPRT0) to reroute the input message to any IMS local or remote destination, as appropriate.
- Builds a message prefix to identify the originating LU6.2 program. This prefix will be propagated throughout the MSC network with the message.
- Queues the transaction to its remote destination on the IMS message queue
- Sends the transaction across the MSC link
- Receives the transaction reply from the MSC link
- Sends this reply either synchronously to the originating LU6.2 application program or asynchronously to any LU6.2 application program.

The remote IMS system provides the following services for the remote application program:

- Receives the incoming transaction from the partner system over the MSC link
- Calls the Link Receive Routing Exit (DFSCMLR0) if provided
- Queues the transaction to its local destination on the IMS message queue
- Schedules the transaction into a dependent region and provides the normal IMS processing facilities (for example, recoverability, rollback and retry, two phase commit).

- Sends back the transaction reply to the local IMS system over the MSC link.
- Keeps the reply on the message queue until an acknowledgment is received.

Program-to-Program Switch

Immediate or deferred program-to-program switching through MSC to a remote application program is also possible. An LU6.2 program can initialize a conversation with a transaction program name of a local transaction, which then issues an ISRT call to a remote destination, either a transaction or an LTERM.

Note: A CPI-C driven application program cannot insert a message to any remote destination, because the MSC message prefix, which defines the originating LU6.2 program, cannot be built.

Message Switch

An LU6.2 program can ask for a message switch to a remote LTERM.

Here is a summary of the DFSAPPC message switch service:

1. From an LU6.2 terminal to an IMS terminal

   ```
   ALLOCATE   TPN=DFSAPPC
   SEND_DATA  (LTERM=lt1) data
   ```

   where lt1 can be a remote LTERM.

2. From an LU6.2 terminal to an LU6.2 terminal:

   ```
   ALLOCATE   TPN=DFSAPPC
   SEND_DATA  (SIDE=si2) data
              or
              (LTERM=lt2) data
   ```

   where si2 is an APPC/MVS Side information member, or lt2 is an LU6.2 descriptor that addresses an LU6.2 destination.

3. From an IMS terminal to an LU6.2 terminal:

   ```
   DFSAPPC (SIDE=si2) data
   DFSAPPC (LTERM=lt2) data
   ```

   where si2 is an APPC/MVS Side information member, or lt2 is an LU6.2 descriptor that addresses an LU6.2 destination.

4. From an IMS terminal to an IMS terminal:

   ```
   lt1  data
   ```

   where lt1 can be a remote LTERM.

Examples 1 and 2 apply when the input is from an LU6.2 program.

Routing Exits

Message routing is either determined by the IMS sysgen transaction definitions or controlled by the routing exits in the IMS systems.

See 6.7, “Input Message Routing Exit (DFSNPRT0)” on page 132 and 6.8, “Enhanced MSC Program Routing Exit” on page 135 for more information about routing.

Message Recovery

Recovering APPC transactions in an MSC environment depends of many factors:

- Resource that fails (LU6.2 session or IMS system or MSC link)
- LU6.2 conversation mode (synchronous or asynchronous):
  - ALLOCATE, SEND, DEALLOCATE for asynchronous conversation
  - ALLOCATE, SEND, RECEIVE for synchronous conversation
- Transaction mode (recoverable or nonrecoverable)

  An APPC transaction is recoverable in an APPC/IMS environment only if INQUIRY=RECOVER is specified on the TRANSACT macro and the LU6.2 conversation mode is asynchronous with sync-level CONFIRM.

  The two TRANSACT macros in the local and remote sysgens must be compatible.
- Transaction type (local or remote).

The following is a summary of the possible sources of failures and their implications. For more information, see the *IMS/ESA V5 Administration Guide: TM, SC26-8014*.

LU6.2 Session Failure: If the failure occurs while IMS is receiving the input message, or before the message gets enqueued on the MSC link, IMS discards the message.

If the failure occurs after the message is enqueued for a remote system, MSC processing continues as normal until the reply is eventually back in the local system.

Whenever a reply is ready to be sent but the session is no longer available, the normal APPC/IMS rules apply:

- For an asynchronous APPC conversation: the output is queued on the DFSASYNC transaction program name.
- For a synchronous conversation: the MCEE is driven (LU6.2 Destination Exit (DFSLULU0) no longer exists in IMS 5.1) to determine the action to be taken.

Discarding a message from an application program execution that has committed updates on the remote site can have serious implications.

MSC Link Failure: Messages enqueued to the MSC link when an MSC link failure occurs are always recoverable. Waiting for MSC link recovery can increase the response time of an LU6.2 synchronous conversation!

Local IMS System Failure: All nonrecoverable messages queued to the MSC links are discarded.

All messages that have been successfully sent across the MSC links will continue to be processed. When the reply comes back to the restarted local IMS system:

- Asynchronous conversation: the output is queued to DFSASYNC transaction program name.
- Synchronous conversation: DFSCMUX0 is now driven to determine the action that has to be taken.

Remote IMS System Failure: All messages that have been enqueued on the local system for the MSC link will remain queued and will be sent when the remote system is restarted.

All nonrecoverable messages in the remote IMS system that have not had their replies enqueued on the MSC link after syncpoint will be discarded. If the discarded nonrecoverable message originated from an LU6.2 synchronous conversation, the conversation will hang because a reply will never be returned. It is then necessary to issue an MVS or VTAM command to deactivate the LU and free the session.

> **Recommendation**: *Define all messages as recoverable in the IMS sysgen of the remote IMS system.*

Distributed Syncpoint Processing

Syncpoint processing is executed only on the one system where the transaction executes. Because MSC involves IMS message queue communication, the transaction is already committed when the output message is sent back to the local IMS system.

When MVS implements distributed syncpoint processing, this situation does not change. Only those applications directly involved in the APPC conversations will have their syncpoints coordinated. MSC links do not use APPC processing.

Security

On the remote system, the remote SYSTEM′s userid is used for RACF security checking (the original userid is probably not defined in the remote system). All remote transactions must be authorized to the userid of the remote control region. Additional security controls can be achieved, however, by using the transaction authorization exit (DFSCTRN0) (if the RACF check succeeds).

During CHNG call processing, the remote security check exit (DFSCTSE0) is invoked regardless of return codes from RACF or DFSCTRN0. Refer to *Remote Security Checking in an IMS MSC Environment (GG66-0291)* for more information.

6.3.3 Implementing MSC for APPC Input Messages

The implementation of MSC for LU6.2 transactions requires careful planning. All IMS systems sending LU6.2 origin or destination messages must be at the IMS 5.1 level—including intermediate IMS systems that do not use LU6.2 directly but are only providing an MSC connectivity function. If any IMS system in the MSC network falls back to IMS 4.1 level from IMS 5.1, LU6.2 traffic cannot flow through the IMS 4.1 part of the MSC network (this may require special attention in the migration plan because messages could be queued on the MSC links for that system on the other IMS 5.1 level systems).

MSC Network

All IMS systems receiving or sending APPC transactions in the MSC network must be at IMS 5.1. This includes all originating systems, the intermediate systems if any, and the remote systems.

Sysgen Changes

MSC links that receive LU6.2 transactions might need larger link buffers to accommodate processing of the LU6.2 prefix. IMS adds a prefix of at least 480 bytes to the LU6.2 message when it is sent over an MSC link. So the BUFSIZE parameter of the MSPLINK macro needs to be increased.

The LRECL for the IMS message queue data sets may also need to be adjusted.

Because the Terminal Routing Exit (DFSCMTR0) is dynamically loaded in IMS 5.1 (if not replaced by the Input Message Routing Exit (DFSNPRT0)), the parameters MSTEXIT and NOMSTEX in the COMM macro of the IMS sysgen are no longer used and should be removed. A warning message is issued during stage 1 processing if they are still present.

6.3.4 Value

Customers who with IMS 4.1 were unable to use native APPC/IMS support because they used MSC will now be able to implement APPC/IMS. The only restriction is that new explicit mode (CPI/C driven) transactions will not be able to exploit any MSC facilities.

An IMS system can be the entry point for all LU6.2 conversations. Only one APPC/MVS system needs to be implemented.

6.4 APPC/IMS Message Mapping Support

APPC/IMS message mapping support addresses two customer requirements that enable an easier implementation of APPC, especially in what was previously a 3270 based customer environment:

- Some customers have written applications that are sensitive to the LTERM name in the IOPCB (for example, the choice of which database to access can be determined by a location code embedded in the LTERM name). The requirement is for IMS to be able to set an appropriate LTERM name in the IOPCB when the input is from an LU6.2 program.
- Some 3270 applications choose from several different output message layouts (by specifying an MFS MODname at IOPCB ISRT time), and MFS handles the screen formatting accordingly. When these same programs are used to process LU6.2 input, a way is needed to tell the remote LU6.2 program which layout has been used for the reply message.

The APPC/IMS message mapping support has been retrofitted to IMS 4.1 with PTFs UN41868 and UN41869.

6.4.1 LTERM Name Support

By default, the IOPCB LTERM field contains the LUname for APPC input. However, an LTERM name can be included in the first SEND of the inbound conversation and be extracted by the DFSLUEE0 LU6.2 user exit. This LTERM name will be placed in the IOPCB. The application program can then use the LTERM name to determine which processing to perform. The nams is also commonly used to determine the destination for any output messages to non-LU6.2 devices such as printers.

6.4.2 MODname Support

MFS support uses a MODname to represent a unique message layout or mapping for an output message.

In a 3270 terminal environment:

- For applications that use MFS for input formatting, the MODname chained off the MID is passed in the IOPCB.
- For the output message, the application can either allow the default MODname (from the IOPCB) to be used for output formatting or change it, using the appropriate parameter on the ISRT or PURG DL/1 call.

Because MFS is not supported with APPC/IMS, the MFS modules are not called. However, IMS 5.1 provides two facilities for exploiting the MODname.

- A MODname specified in the first SEND of the input to IMS (instead of or as well as the LTERM name described above) can be extracted by the LU6.2 Edit Exit (DFSLUEE0) and placed in the IOPCB.
- The MOD name used by an IMS application ISRT (default or as set by the program) can be included in the first SEND to the remote LU6.2 program. Again this function is performed by the LU6.2 Edit Exit (DFSLUEE0).

These facilities provide several potential benefits:

- Applications originally designed with 3270 in mind are fully compatible with LU6.2 processing.
- The LU6.2 remote application can be told explicitly which message mapping has been used.
- The LU6.2 program can also more easily handle asynchronous replies.

6.4.3 Use of ETO Initialization Exit

The Initialization User Exit (DFSINTX0) no longer requires that ETO be included in the IMS sysgen. This user exit is now available in all IMS TM environments. DFSINTX0 can be used to create two user tables, one for ETO and one for APPC/IMS usage. Typically, each of these tables is a table of pointers to other tables created or loaded by the exit. DFSINTX0 is called at IMS TM control region initialization time. The APPC/IMS tables can contain any information that is useful to the APPC/IMS exits. With respect to the LTERM and MODname support, these tables might contain LU to LTERM mapping, trancode to MODname mapping, and the like. In this way, the remote LU6.2 program does not need to be involved in setting an LTERM and/or MODname in the first SEND of the input data.

6.4.4 Functional Flow

Figure 31 on page 129 shows a sample implementation of APPC/IMS message mapping support.

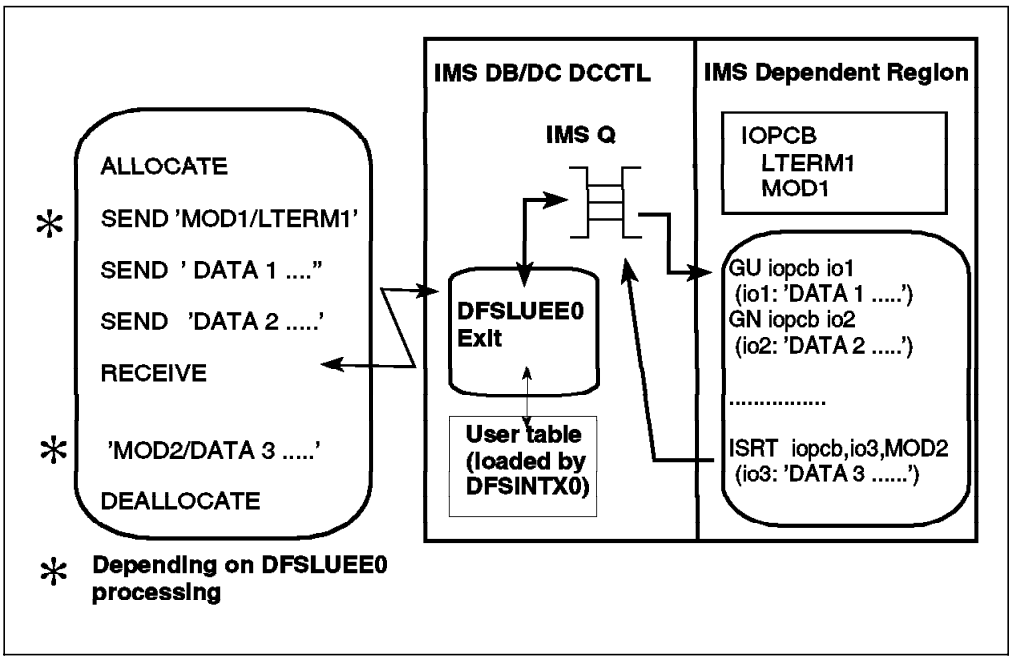


Figure 31. APPC/IMS Message Mapping Support

In this example, the steps are:

1. The LU6.2 application program sends both an LTERM and a MODname in the first segment of the input message.

   The format of this first segment is mutually defined by the remote programs and the LU6.2 Edit Exit (DFSLUEE0). (Either or both fields can be sent.)

2. DFSLUEE0 is used to:

   - View and change the contents of a message segment before storing it in the message queue
   - Discard a message segment (appropriate in this example)
   - DEALLOCATE_ABEND the LU6.2 conversation.

3. For the input message, DFSLUEE0 checks the contents of the first message segment, consults the user tables loaded by the Initialization User Exit (DFSINTX0), and decides whether to pass the LTERM and/or MODnames to IMS in the IOPCB.

4. For the first segment of the output message, IMS provides the MODname to DFSLUEE0, which can format the message accordingly or simply include the MODname in the output message sent to the LU6.2 partner.

6.4.5 Value

Customers migrating 3270 systems onto APPC platforms may find that the LTERM and MODname support feature of IMS 5.1 allows them to avoid having to change IMS application programs. LTERM dependencies and multiple output message layouts can be handled with very simple APPC code in the workstation (an additional SEND and/or RECEIVE).

6.5 APPC/IMS Network-Qualified LUnames

A network-qualified LUname consists of a one- to eight-character network identifier of the originating system followed by a period and the one- to eight-character name of the LU within the network. Without full qualification, network LUnames must be unique within the local and the interconnected networks. With full qualification, network LUnames must be unique only within a given network.

APPC/MVS allows the use of LU6.2 fully qualified LUnames to allocate an inbound conversation. But for outbound conversations, APPC/MVS strips off the network ID and only passes the LUname to VTAM. Therefore the LUname must still be unique within the local and all interconnected networks if outbound conversations are to be used to nonlocal networks.

6.5.1 Implementation

Using network-qualified names is often straightforward. This change only applies if it is mandated by the present and future operating environment of the IMS system. In many cases the implementation can be deferred indefinitely.

IMS Commands

In all appropriate IMS commands, the LUname can be network-qualified. Here is a list of all commands where the LUNAME keyword can be used:

```
/ALLOCATE                    /START
/CHANGE DESCRIPTOR           /STOP
/DEQUEUE                     /TRACE
/DISPLAY
```

The fully qualified LUname must be surrounded by single quotes. If they are missing the period is interpreted as the end of the command input.

AOI Utilization

If an IMS command with the network-qualified LUname is passed to either of the automated operator exits (DFSAOUE0 or DFSAOE00), IMS modifies the LUname in the input command before passing it to the exit routine.

The command:

```
/DISPLAY LUNAME 'NETWORK1.LUNAME1'
```

is passed to the AO exit routines or logged to the secondary master as:

```
/DISPLAY LUNAME NETWORK1:LUNAME1
```

IMS Calls

In the CHNG call, the LUname can be network-qualified.

For the INQY NULL call, a network-id field is added in the output area. This implies an 8-byte increase in the IOAREA size.

LU6.2 Descriptor, DFSAPPC, and Exit Support

The LUname in the LU6.2 descriptor can be network-qualified. This fully qualified LUname can also be used in DFSAPPC message switch services, in DFSLUEE0, and in the MCEE.

6.5.2 Migration Considerations

Because network-qualifier LUnames use is still optional, there is no migration implication.

6.5.3 Value

Subject to the APPC/MVS outbound conversation limitation, a unique name for every LU6.2 LU on every system in a network is no longer needed.

6.6 MSC Routing Exits

In general, before to IMS 5.1, the MSC Routing Exits provided significant function but not much flexibility. LTERMs and transactions in one system typically needed to be defined in some or all of the other IMS systems. For example, to route an input message to a different system required that the destination be set to a transaction that was defined, in the originating system, as a remote transaction. Directed Routing is an MSC facility that can reduce the need for having unique names across all systems, but it requires user applications to participate in the routing process.

With the advent of the Parallel Sysplex, some customers may need a greater number of connected IMS systems and might choose to have a basic system definition from which all other systems are cloned. Although the MSC link definitions must be uniquely defined in each IMS system, an objective of the IMS 5.1 MSC enhancements is to make it possible to remove the need for having unique transaction and LTERM names in each system. The MSC routing exits will, where appropriate, specify the destination system and may not need to change the transaction code or LTERM.

MSC Exits before IMS 5.1

Before IMS 5.1 the MSC routing exits were:

- Terminal Routing Exit (DFSCMTR0)

  DFSCMTR0 was called when a message arrived from a terminal. It could set the destination to any local or remote transaction known to the originating IMS system. Additionally, it had the option of changing the transaction code in the message.

- Link Receive Routing Exit (DFSCMLR0)

  DFSCMLR0 was called when a message arrived from a remote system through MSC. Based on the current destination transaction and the actual transaction code in the message, it could set a different destination transaction.

- Program Routing Exit (DFSCMPR0)

  DFSCMPR0 was called when a program, processing a message that originated in a different system, issued a DL/1 CHNG call. Based on the originating IMS ID and LTERM and the CHNG call destination, it decided whether to reject the CHNG (A1 status code), let the current system handle

the message, or send the message to the originating system for final destination determination. To some degree, this allowed the same LTERM names and transaction codes to be defined in multiple systems.

MSC Exits in IMS 5.1

In IMS 5.1, there are enhancements that affect the Terminal Routing and Program Routing exits. These are described in the sections that follow.

The Terminal Routing Exit is functionally replaced by a new general purpose routing exit, the Input Message Routing Exit (DFSNPRT0). A major difference is that DFSCMTR0 does not handle APPC/IMS or OTMA input messages, but DFSNPRT0 does. Also, the new destination does not have to be defined on this originating system. The exit can specify the destination system, using either the MSNAME or the SYSID.

The Program Routing Exit has been enhanced and is called on CHNG calls regardless of whether the message originated in the local or a remote system (see 6.8, “Enhanced MSC Program Routing Exit” on page 135 for more specific details of when the exit is called). As well as choosing the local or originating system, it can now set any other connected system as the destination system.

Figure 32 on page 133 shows the MSC user exits available in IMS 5.1 for message routing.

6.7 Input Message Routing Exit (DFSNPRT0)

This new MSC user exit replaces the MSC Terminal Routing Exit (DFSCMTR0). The old exit can still be used, but only if DFSNPRT0 is not used. A conversion to the new exit is recommended.

This exit is functionally equivalent to the MSC Terminal Routing Exit (DFSCMTR0). The difference is that the new exit will be called for LU6.2 and OTMA input messages as well as non-LU6.2 messages.

DFSNPRT0 is a new general-purpose input routing exit. It is not specified at IMS generation on the COMM macro and it does not require MSC to be specified in the IMS system. Instead, IMS tries to load the exit at startup time from a RESLIB pre-concatenation. If the exit is found, it will be used.

The MSC Terminal Routing Exit (DFSCMTR0) can still be used, but only when DFSNPRT0 does not exist. It is no longer linked in with the IMS nucleus. IMS issues a load request from the RESLIB concatenation at startup time, and, if DFSCMTR0 is found, it will be enabled.

The COMM macro OPTION=MSTEXIT is no longer required.[9]

If used, the MSC Terminal Routing Exit remains unchanged and will not be called for LU6.2 or OTMA messages.

The new Input Message Routing Exit (DFSNPRT0) must be written in assembler, be reentrant, and be capable of running in 31-bit mode.

[9] IMS generates a warning message at Stage 1 time if either MSTEXIT or NOMSTEX is specified.

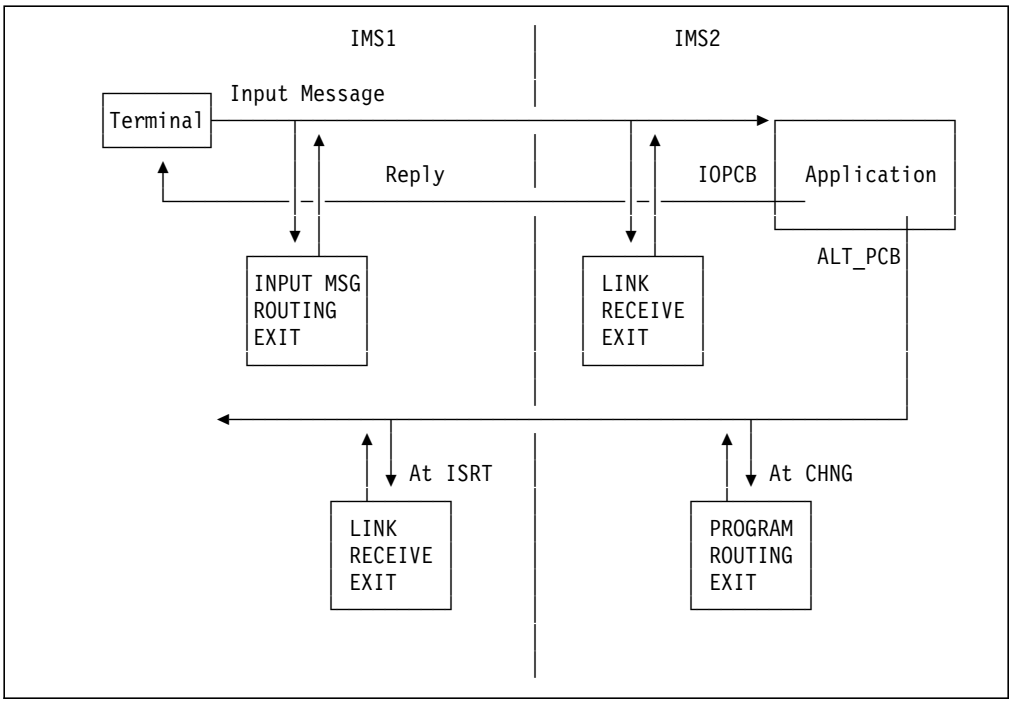


Figure 32. MSC Routing Exits in IMS 5.1

DFSNPRT0 will be called when the first segment of a new input message arrives from any source. In the case of IMS conversational transactions (that is, those that use a SPA), the exit is only called for the first segment of the message that starts the conversation.

The purpose of the new exit, like DFSCMTR0, is to set a new destination and optionally change the trancode in the message. The new destination can be local or remote, LTERM or transaction, if the message originated at a traditional (not LU6.2 and not OTMA) terminal. For LU6.2 and OTMA, the destination can only be set to a transaction (local or remote).

DFSNPRT0 can use Callable Services, but only when processing a message from a non-LU6.2, non-OTMA terminal.

When DFSNPRT0 is called, as with all of the newer exits, register 1 is used to pass the address of a standard parameter list that contains the address of the exit-specific parameter list (see Table 9 on page 134).

Table 9. Exit-Specific Parameter List for DFSNPRT0			
Offset	**Length**	**Description**	**Set by Exit**
Common Parameters			
0	4	Current ECB address	
4	4	Message flag 0 Message originated from non-LU6.2/non-OTMA terminal 4 Message originated from LU6.2 program or device 8 Message originated from OTMA	
8	4	Address of first message segment	
12	4	Address of 1- to 8-byte destination field	√
28	4	Address of 8-byte MSNAME or SYSID	√
16	4	Length of destination name	√
LU6.2-Specific Parameters			
20	4	Address of 1- to 17-byte input LUname	
24	4	Address of LU6.2 Message Information Block This is mapped by DSECT DFSMSGRT and contains the LUname, userid, group name, and transaction program name.	
OTMA-Specific Parameters			
20	4	Address of 8-byte Tpipe name	
24	4	Address of 16-byte member name	
Other Terminal Types			
20	4	Address of 8-byte LTERM name	

To set a different destination from the trancode in the message, the exit can either change the destination at the address at offset 12 or change the address at offset 12 to point to a field containing the new destination. The value at offset 16 must also be set to the length of the destination name. Offset 28 is used to specify the address of a field containing either an MSNAME or a SYSID.

6.7.1 Migration Considerations

The following changes in the MSC user exits are recommended even if the new functionality is not immediately utilized:

DFSCMTR0

If still using DFSCMTR0, note that the COMM macro should have the OPTIONS=MSTEXIT removed (to avoid a warning message at IMS sysgen time) and the module itself should no longer be link-edited with the IMS nucleus.

DFSNPRT0

DFSNPRT0 is meant to eventually replace DFSCMTR0. Consequently, it is recommended that customers currently using DFSCMTR0 plan to migrate to DFSNPRT0. Customers wanting to use MSC with LU6.2 and/or OTMA must convert if they use the DFSCMTR0 exit today.

DFSNPRT0 will be used whenever it is in a library concatenated before RESLIB.

A sample DFSNPRT0 can be found in the IMS TMSOURCE library.

6.8 Enhanced MSC Program Routing Exit

Existing user exit coding for DFSCMPR0 should be carefully reviewed even if the new exit capability is not utilized, as this user exit is now called (potentially for transactions that do not use MSC, and which would not have been visible to the user exit in earlier releases).

The Program Routing Exit has been enhanced to be called on local as well as remote systems, and several more routing options have been made available.

The Program Routing Exit is called when an application issues a DL/1 CHNG call. Before IMS 5.1, the exit was called only when the application was executing on a remote system, and its options were (a) let the processing system interpret the CHNG call destination, (b) route the message to the originating system for destination determination, or (c) reject the CHNG call with an A1 status code.

In IMS 5.1, the Program Routing Exit is called in new circumstances and has a larger ranger of routing options. It is called for an alternate PCB CHNG call when the following conditions apply:

- MSC is generated in the system (at least one PLINK macro)
- Application is a nonconversational MPP or BMP
  - Application can be local or remote
- A GU against the IOPCB has successfully completed
- CHNG call PCB is not an alternate response PCB.

The exit recognizes whether or not the CHNG call is being processed on the originating system by the value in register 0:

R0	**Meaning**
Positive	SYSID of originating system if CHNG call on remote system
Negative	SYSID when processing system is the originating system

The routing function is determined by setting the return code in register 15. The possible values and their meanings are shown in Table 10 on page 136.

Table 10. MSC Program Routing Exit: Routing Options

Return Code	Routing Option
00	Use destination set by CHNG call on the processing system
04	Route message back to originating system, using the destination *LTERM* set by the CHNG call
08	Reject CHNG call (A1 status)
0C	Route message back to originating system, using the destination *transaction* set by the CHNG call
10	Route the message to a specified MSC system and use the *LTERM* specified in the CHNG call The target system is defined by use of registers 1 and 2: • Routing by SYSID **Register** **Contents** **0** SYSID (positive, nonzero value) **1** Not used • Routing by MSNAME **Register** **Contents** **0** Zero **1** Address of 8-byte MSNAME
14	Route the message to a specified MSC system and use the *transaction* specified in the CHNG call The target system is defined by use of registers 1 and 2: • Routing by SYSID **Register** **Contents** **0** SYSID (positive, nonzero value) **1** Not used • Routing by MSNAME **Register** **Contents** **0** Zero **1** Address of 8-byte MSNAME

6.8.1 Implementation

The way this exit is defined and loaded has not changed. The COMM macro OPTIONS=MSPEXIT is still required, and the module should be linked with the IMS nucleus.

6.8.2 Migration Considerations

DFSCMPR0 from a previous release will almost certainly need to be modified to work with IMS 5.1 because it is called for all CHNG calls (except in Fast Path EMH programs), not just those on remote systems. The exit should at least be updated to check for register 0 being negative; if it is negative, RC=0 in register 15 should be set.

6.9 Value of Enhancements

The terminal input routing function used to be available only to MSC users but has now been made available to all IMS transaction manager users. Should there be a business or operations reason for segregating some messages from others, for example, the Input Message Routing Exit can perform this function.

In a Parallel Sysplex environment, it might be necessary to route certain transactions to a particular system. The enhancements in IMS 5.1 make this possible without requiring the definitions of remote transactions, and without needing applications to be changed to use directed routing.

Overall, the restrictions in previous releases have been removed, resulting in greater flexibility and ease of use.

Chapter 7. IMS 5.1 Miscellaneous Enhancements

This chapter describes those enhancements that do not fall into any of the previous categories. It also includes the functions that are no longer supported in IMS 5.1.

7.1 Enhanced Database Access Security

In this section, the many references to RACF should be interpreted to mean "RACF or an equivalent security product."

Before IMS 5.1, online IMS systems (TM or DBCTL) did not perform any authorization check when opening a VSAM data set (full function DL/1 or Fast Path DEDB). Online IMS systems invoked RACF when opening an OSAM data set and used RACF for all databases in offline batch. This anomaly is addressed in IMS 5.1, which now always performs a security check at data set open time for all database types in all environments.

Many customers have defined all of their database data sets with RACF protection. However, before IMS 5.1, a RACF check was not always made at data set open time, as might have been expected. Table 11 shows the situation in earlier releases.

Table 11. Use of RACF at Database Open in Releases before IMS 5.1

	Full Function VSAM	**Fast Path DEDB**	**OSAM**
Online	No RACF check (regardless of PPT option)	No RACF check (regardless of PPT option)	RACF checked (if not disabled by PPT option)
Batch	RACF checked	Not applicable	RACF checked

In IMS 5.1, RACF is called at open time for all database data sets in any environment. It is not an option, although it is possible to disable *all online* IMS data set open RACF checking by updating the MVS Program Properties Table (PPT) entry for DFSMVRC0. For example, member SCHEDxx in MVS Parmlib should contain:

```
PGMNAME(DFSMVRC0) NOSWAP SYST KEY(7) NOPASS
```

Note that this applies to all online IMS systems on this MVS image.

Table 12 shows the userids used with IMS 5.1 at data set open time for RACF checking.

Table 12. Userids Used for RACF Check at Database Open in IMS 5.1

	Full Function VSAM	**Fast Path DEDB**	**OSAM**
Online	DL/1 SAS userid, if assigned Control region userid if no SAS userid	Control region userid	DL/1 SAS userid
Batch	Batch job userid	Not applicable	Batch job userid

If a relevant userid (or userids in the case of full function VSAM) has not been assigned, and the data set does not have universal access defined in RACF, IMS will not be able to open that data set.

7.1.1 Implementing RACF Database Protection

The IMS system address spaces are typically run as started procedures (STCs). The customer's Security Administration group should be contacted to have the RACF userids assigned in the started task table (ICHRIN03). For simplicity, the same userid should be assigned to both the control region and the DL/1 SAS.

7.1.2 Migration Considerations

Because online authorization was not requested for VSAM databases (including DEDB) in earlier versions of IMS, it would be a wise precaution before moving to IMS 5.1 to check that the database data sets are defined to RACF, the IMS control region and DL/1 SAS have been assigned userids, and these userids are authorized to access the data sets.

7.1.3 Value

All customers with high data security requirements will benefit from this RACF enhancement. IMS 5.1 provides the facilities to fully control which database data sets can be authorized for use by each of the online systems. The previous security exposure in certain MVS system configurations has been removed.

7.2 Up to 999 Dependent Regions or CICS Threads

For some customers, the limit of 255 dependent regions or CICS threads to DBCTL has proved to be a problem. In IMS 5.1, the limit has been increased from 255 to 999 PSTs. Each PST represents a single application unit of work, either a CICS thread or IMS dependent region. This change, combined with n-way data sharing, makes it possible for up to 31,968 dependent regions or CICS threads to have access to the same data! IMS is committed to, and capable of, massively parallel computing on a significant scale.

Two types of customer have already recognized the 255 limit:

- DBCTL users with multiple CICS systems accessing the same DBCTL system

  For example, with a maximum number of threads defined as 20 for each CICS system, DBCTL cannot support more than 12 CICS systems.

- Users of APPC/IMS with SYNC_LEVEL=CONFIRM and synchronous conversations

  At (actually before) syncpoint, the IMS partner application waits for message delivery to the "terminal" and return of the resulting CONFIRMED. In other words, the dependent region is waiting on network communication flows, which, in comparison to host processing and I/O, can be very long. Region occupancy is much higher and more regions are needed to support the same transaction rate.

IMS 5.1 introduces a new control region execution parameter, `MAXPST=`, which specifies the maximum number of dependent regions or threads allowed. The default value is 255.

There has been no change to the other JCL parameter, PST=, and the IMSCTRL macro parameter, MAXREGN=. Both parameters define a minimum number of regions or threads for which IMS will provide control blocks. When the number of regions increases beyond this specified limit, more control blocks are dynamically created.

The MAXPST= parameter also addresses another need. Previously there was no definable maximum for the number of dependent regions or threads. Operators could start more regions (up to the old 255 limit) and cause the database pools, PSB pool, and VSAM strings, for example, to become overcommitted. The MAXPST= parameter can be used to limit the number of regions the operators can start.

7.2.1 Implementing More Than 255 PSTs

Increasing the number of dependent regions or threads should be done only if necessary, and then only after careful planning. The new maximum of 999 is intended to be much greater than any customer should require in the near future.

The system aspects that must be considered have to do with the resources that the dependent regions use. This includes the sizes of the database buffer pools (VSAM, OSAM and Fast Path common pool), the PSB and PSBW pools, and the PI ENQ/DEQ pool (PIMAX).

Even if the reason for having more regions is that many regions are waiting on APPC, it is still important to ensure that programs that are ready to execute do not have to wait unnecessarily. Therefore, the number of VSAM strings should be increased accordingly (STRINGNM parameter in DFSVSMxx).

Also, if significantly more virtual storage is to be used, there will be a need for correspondingly more real storage.

7.3 Four-Digit Device Addresses

This change primarily affects BTAM devices. Although still supported by IMS, it is hoped that few BTAM devices are actually in use. In particular, any MSC links using CTC or BTAM should be converted to VTAM for operational and availability reasons.

MVS version 5 allows more than 4096 logical units and so supports device addresses of up to 4 hexadecimal digits. IMS 5.1 has been modified to support four-digit addresses.

The IMS 5.1 support of four-digit device addresses is almost completely transparent. Two Stage 1 SYSGEN macros can be affected, and certain /DISPLAY commands allow for four digits in certain fields.

The two macros that can use four-digit addresses are:

```
MSPLINK  ADDR=
LINE     ADDR=
```

The ADDR parameter on an MSPLINK macro is applicable for IMS CTC (and BSC) MSC PLINKs. The LINE macro applies only for BTAM terminals.

The /DISPLAY commands that are affected are those that reference lines and MSC links. For example:

Command:

```
/DISPLAY ASSIGNMENT MSPLINK ALL
```

Response:

```
LINK PLINK    TYPE  ADDR     MAXSESS  NODE
   1 PLINKLU6 VTAM  ****       001    LUA1      PSTOPPED
   2 PLINKLU6 VTAM  ****       004    LUB1
   3 PLINKLU6 VTAM  ****       004    LUB1
   4 PLINKLU6 VTAM  ****       004    LUB1
   5 PLNKCTC1  CTC   370
   6 PLNKBTAM  BSC  10BA
   *94244/093155
```

Only one message is changed (the UNIT= value):

```
DFSO762I OSAM (TAPE|DASD) (READ|WRITE) ERROR - FUNC=aa
         STATUS=bb,cc,dddd,eeee,ffff
DFSO762I OSAM UNIT=gggg FAILING CCW=hh,,ii,jjjj IOSEEK=kkk...kk LOG=lll
DFSO762I OSAM DSN=..............
```

7.4 Change to /DISPLAY AREA Command

The new /DISPLAY AREA command reduces the amount of I/O caused by the command and shortens the response time for receiving the command response. An option provides for executing the old form of the command.

In “POS Call and IOVF Statistics” on page 48 we explain that the count of available IOVF CIs is no longer maintained in storage but is measured by reading all of the space maps. In previous releases of IMS, this value was included in the response to a /DISPLAY AREA command.

In IMS 5.1, an additional command parameter, IOVF, is required if the IOVF statistics are to be shown. The default is not to calculate the number of free CIs.

/DIS AREA DEDBAR03 IOVF

```
AREANAME  EQECNT   TOTAL UNUSED  TOTAL  UNUSED     DBNAME CONDITIONS
 DDNAME   REMAIN  SEQ DEPENDENT DIR ADDRESSABLE
DEDBAR03    N/A      342    115    172    108-CI DEDBDB02 VIR, PREO, PREL
 DBAR03A1    10      N/A    N/A    N/A    N/A
```

/DIS AREA DEDBAR03

```
AREANAME  EQECNT   TOTAL UNUSED  TOTAL  UNUSED     DBNAME CONDITIONS
 DDNAME   REMAIN  SEQ DEPENDENT DIR ADDRESSABLE
DEDBAR03    N/A      342    115    172    N/A    DEDBDB02 VIR, PREO, PREL
 DBAR03A1    10      N/A    N/A    N/A    N/A
```

7.4.1 Implementation

Unless this command is used specifically to retrieve IOVF space statistics, no change to operating procedures is necessary.

7.5 Reduced Logging at System Checkpoints

The impact of IMS system checkpoints is a concern for a few systems (primarily very large systems and/or systems with very high transaction rates). IMS 5.1 has reduced the amount of log data written at system checkpoints and thus the impact of this checkpointing activity. The enhancements are specific to system checkpoints; they are not related to application checkpoints taken with the DL/1 CHKP call.

Reduced logging at system checkpoints was retrofitted to IMS 4.1 and made available by APARs PN42403 and PN48944.

IMS 5.1 (and IMS 4.1 with the fixes applied) no longer logs complete control blocks in some cases, only the essential data from those blocks. Table 13 shows the control blocks for which checkpoint log data reduction applies and the reduction in log data per block.

Table 13. IMS 5.1 Checkpoint Log Reduction

Control Block	Log Reduction	Log Record
Static LTERM (CNT)	56 bytes	′4003′
ETO LTERM (CNT)	56 bytes	′4014′
USER (SPQB)	20 bytes	′4014′
CCB	28 bytes	′400D′

7.5.1 Value

A user with 10,000 ETO signed-on users with 5000 conversations would experience a total log reduction of almost 1MB per checkpoint.

7.6 Removed Functions and Support

The functions discussed below are no longer supported in IMS 5.1 and should not be used.

7.6.1 CICS Local DL/1

The code that supported CICS local DL/1 has been removed from the IMS 5.1 Database Manager product. CICS DL/1 users must convert to using DBCTL before using IMS 5.1.

7.6.2 LU6.1 Adapter for LU6.2 Applications

The LU6.1 adapter for LU6.2 applications is no longer supported in IMS 5.1. The code is still shipped and should function with IMS 5.1. However, there is no program service support for problems encountered with the adapter on IMS 5.1.

All users of the adapter should migrate to APPC/IMS before implementing IMS 5.1. This may not be possible in some cases, for example, if using MSC, in which case the migration can be done either as part of the IMS 5.1 migration or afterward.

Migration from LU6.1 Adapter to APPC/IMS

Applications written to use the IMS LU6.1 adapter are upwardly compatible with APPC support. However, certain changes may be necessary, namely:

- Use RACF (or equivalent product) to provide security directly, rather than through the IMS /SIGN ON command
- The TPN to which IMS sends unsolicited and asynchronous output messages is DFSASYNC instead of IMSASYNC
- The TPN to which IMS sends command responses is DFSCMD.

Benefits of APPC/IMS

APPC/IMS removes some restrictions in the support of LU6.2 protocols. Migration is also important because the LU6.1 adapter has serious data integrity exposures that can cause lost or duplicate messages. The adapter does not have the integrity characteristics of the rest of the IMS system (it does not have a log).

Parallel Session Support: APPC/IMS can have thousands of sessions to the same LU6.2 device. This support provides much greater throughput capability than the adapter and greatly improves the viability of using LU6.2 to connect IMS to CICS systems.

The adapter does not support parallel sessions. This is a severe operational problem with multiuser software, such as CICS, AIX, or VM.

Length of Data: With APPC/IMS, there is no limit to the amount of data (either length or number of sends) that can be received. The input message can consist of any number of message segments; each segment can be of any length up to the maximum supported by the IMS queue manager (for current or modified application programs).

To minimize the risk of a message queue overflow when a runaway application is inserting very large messages, the Queue Space Notification Exit (DFSQSPC0) is provided to control the space in the message queue data sets assigned to a specific application program. This exit can then deallocate the LU6.2 conversation.

CONFIRM and SEND_ERROR Verbs: These verbs are fully supported by APPC/IMS for both inbound and outbound usage, including implementation of their intended semantics.

The adapter does not support this part of the LU6.2 protocol. An error occurs, with possible data integrity exposures if these flows are accidentally used by an application using the adapter.

Synchronous Conversation: The adapter does not support synchronous conversations because the LU6.1 protocol has only asynchronous capability. The adapter maps the LU6.2 synchronous conversation onto the LU6.1 asynchronous message flow, but in so doing the result is the same as asynchronous for LU6.2.

APPC/IMS supports synchronous conversations, and this support is visible to the partner LU6.2 program.

Chapter 8. The IMS/ESA 5.1 System

This chapter brings IMS V5.1 into true perspective and shows why it deserves to be called the “best of breed” transaction processor for MVS.

Many customers, worldwide, have “bet their business” on the IMS Transaction Manager, which provides:

- Robustness
- High availability
- Excellent performance
- Low cost per transaction
- Uncompromising integrity (for data and messages)
- Protected investment in many years of application development

In this chapter we examine:

- Some of the features that contribute to IMS′s superb reputation
- Where IMS is currently positioned in the rapidly changing world of information technology
- Where IMS is moving in the future, especially in relation to the open systems environment.

8.1 IMS Performance

The IMS Transaction Manager is noted for its high performance and low cost per transaction. Clearly, performance is most apparent when using IMS Fast Path, but full function message processing with access to DB2 data still achieves very high levels of performance.

IMS TM performs so well for many reasons, some of which are summarized in the sections that follow.

8.1.1 Exploiting Data In Memory

IMS is a system that contains many pools of buffers and control blocks. During the past 10 years, these pools have been systematically made larger to exploit the growing processor sizes and minimize the amount of system I/O required to support the transaction processing.

Database buffer pools can be very large, and MVS hiperspace extensions can be defined for VSAM pools. You can have many pools and allocate different sets of databases to different pools. Also, separate sets of buffers can be allocated for use only by indexes.

The **message queue pool** can also be made large enough to eliminate queue I/Os except for exceptional cases (for example, stopped transactions, paper jams on printers). Special techniques have been devised to ensure that a large queue pool size does not impact system checkpoint performance.

IMS **conversational processing** has been made very efficient by eliminating all SPA I/Os. The **SPA pool** grows and contracts dynamically according to system

demand. At least one IMS customer has 5,000 active conversations with a SPA size of 10KB (50MB in total).

IMS 5.1 introduces the **DEDB VSO option**, which enables DEDB areas to be kept in virtual storage, thus eliminating both read and write application I/Os.

8.1.2 Parallel Processing

The IMS Transaction Manager is a multi-address-space subsystem (more than 1000 address spaces in theory). It also uses multitasking within address spaces, and especially so in the IMS control region. The degree of parallelism in the system depends on distributing the transaction processing work across the available tasks, and in IMS 4.1 a significant improvement was made in taking work, previously serialized through an IMS control task, and sharing it across the dependent regions. APPC/IMS support, also introduced in IMS 4.1, was specifically designed to use multiple tasks within the control region.

Another benefit of the multiple address space structure with each application program running as a separate task is when an application program accesses DB2. The IMS application program can execute either DL/1 or DB2 code without having to switch to a different task and thus avoids all of the switching and dispatching overheads. It is also worth noting that in IMS 5.1 the process of each application program gaining access to the DB2 subsystem has also been made more efficient.

IMS databases have also seen a growth in parallelism. At each application commit point (syncpoint), OSAM and VSAM updates are done in parallel, and multiple OSAM data sets are themselves updated in parallel. For Fast Path, the DEDB updates are completely asynchronous and exploit parallelism within themselves.

8.1.3 Smart Processing

IMS has benefitted from many design changes that allow functions to be performed more efficiently, so adding to the overall performance. For example, as already mentioned, system checkpoint processing has been made more intelligent to avoid unnecessary work. PSB cloning is exploited whenever possible to avoid reading from the ACB library. Low activity but essential terminal control blocks can be paged out to expanded storage (but no further) and still be readily available.

8.1.4 Logging

Many IMS activities have to be logged, and some events must wait for logging before they are allowed to happen (log write ahead). It is essential, then, in a high performance system, that the logger runs smoothly and efficiently. This is certainly the case for the IMS logger, which is probably the best logger in the industry today. Not only does the IMS logger provide complete integrity of the IMS resources and achieve excellent performance, but it also manifests first class availability characteristics.

8.1.5 Parallel Sysplex

Customers today run IMS systems that range in size from small to large to huge. At the high end, some customers see a continuing growth in transaction volumes that will be impossible to meet with a single large processor. Regardless of IMS system size, almost all IMS customers are looking to further reduce the cost of each transaction.

The IBM solution to both requirements is the Parallel Sysplex. This allows many MVS systems to be loosely coupled but provide a single system image to the end user. The MVS systems can be run on the traditional bipolar processors (ES/9000s), in which case the aggregate processor power available becomes very considerable.

Alternatively (or in addition), the MVS systems can be run on the new CMOS based processors, which provide computing power at significantly reduced cost.

In both cases, the MVS architecture is the same. IMS is now evolving to fully participate in this new MVS Parallel Sysplex architecture. IMS 5.1 introduces Parallel Sysplex support for IMS databases, often referred to as n-way sharing. With IMS 5.1 it is possible to run multiple IMS TM, CICS DBCTL, and batch systems across multiple MVS systems, all reading and updating the shared databases efficiently and with integrity.

A subsequent release of IMS will provide exploitation of the new MVS architecture by the IMS Transaction Manager. The IMS message queues will exploit the Coupling Facility and be a single sysplexwide resource at the heart of the IMS sysplex. Logically, the function of the IMS systems can be considered to be split into network driving and transaction processing. Network driving IMS systems will be receiving messages from the various networks (SNA and non-SNA) and placing them on shared message queues. Transaction processing IMS systems will be taking messages off the queues and inserting replies back to the queues. The network driving IMS systems will then take the replies from the queues and send them to their final destinations. This conceptual view is illustrated in Figure 33 on page 148.

8.2 IMS Availability

IMS is well known for its high availability characteristics. There are several aspects of availability for which this applies:

- For most IMS TM users, IMS system failures are exceedingly rare. But the concept of availability must also encompass the individual components of the system (such as application programs, databases, links, and terminals).
- Maintaining high availability is not just about avoiding failures; it is also about avoiding scheduled outages.
- The pragmatic view of availability recognizes that, regardless of the precautions are taken, failures or outages will occur. The important point is to minimize the duration of the loss of availability.

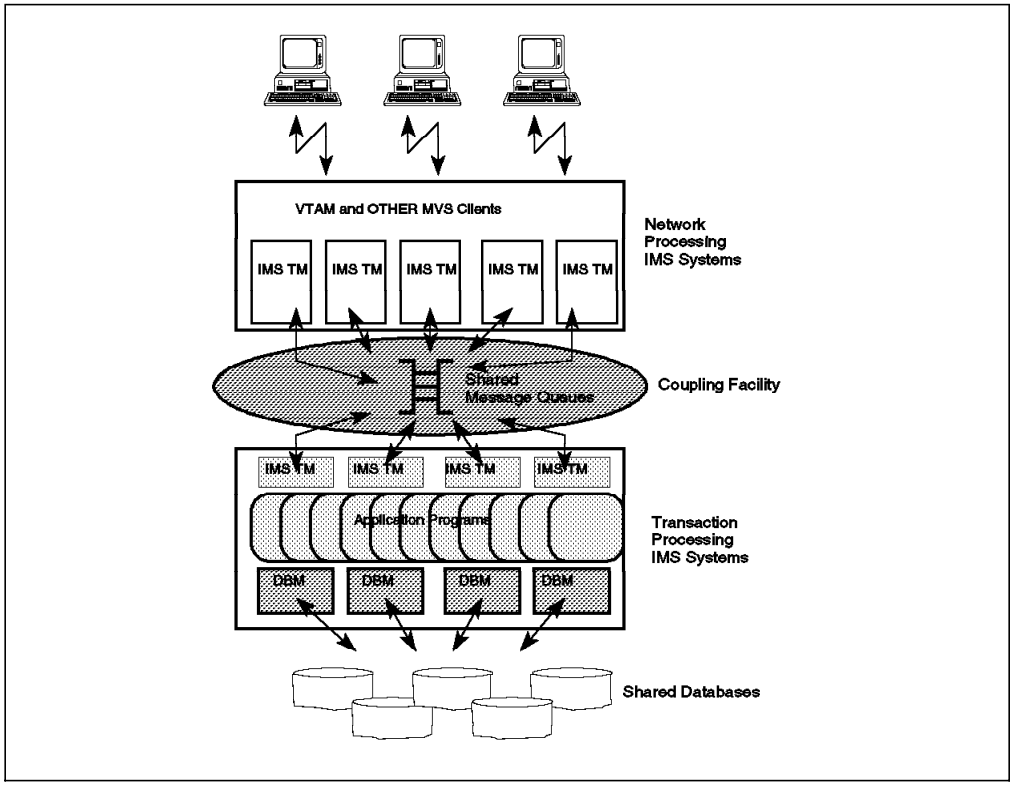


Figure 33. Conceptual View of an IMS TM Parallel Sysplex of the Future

8.2.1 Unscheduled Outages

IMS was the first MVS program to exploit hot standby. Using **XRF**, one IMS system shadows another, and takes over within seconds if the active system fails. With XRF, a number of IMS TM customers worldwide have achieved continuous service availability in excess of 1000 days.

In IMS 5.1, the concept of a standby system is extended from a local perspective to **RSR**, which provides system managed transfer and application of log data to shadow databases at the remote site. A key feature of RSR is that the remote databases are assured of consistency and integrity. In the event of a site failure, the remote system can be restarted as the production system within minutes.

8.2.2 Scheduled System Outages

The **online change** facility in IMS TM allows almost all user elements of an IMS system (such as transactions, databases, screen formats, and security) to be changed without stopping IMS. The major exception is terminals, but this is addressed by the **ETO** feature. Introduced in IMS 4.1, ETO provides several benefits:

- Terminals no longer need be predefined in the IMS sysgen.
- Necessary control structures are built dynamically at LOGON and SIGNON times.
- Comprehensive printer support is included.

IMS 4.1 also introduced **dynamic pool management** whereby many buffer pools grow and contract as required, so avoiding the need to stop IMS just to increase a pool size.

The DBRC PRILOG record used to grow ever larger as an IMS system archived more and more logs. Again in IMS 4.1, a dynamic **PRILOG compression** facility was introduced.

Some IMS outages are scheduled for reasons unrelated to IMS. But IMS can help avoid them. For example, machine powerdowns can be addressed by XRF. Similarly, in IMS 5.1, computer center powerdowns can be addressed by the RSR feature.

8.2.3 Scheduled Data Outages

Fast Path DEDBs are still unique in providing online database reorganization. In IMS 5.1, the online reorganization utility runs many times faster than in previous releases. The DEDB area replication facility includes an online utility to dynamically add extra copies, and this utility can be used to effectively provide both online recovery and an online data set move facility.

The Concurrent Image Copy utility applies to full function and Fast Path databases and allows databases to be copied quickly and efficiently without ever being taken offline.

8.2.4 Component Failure

IMS provides some powerful and unique features that increase component availability in many different ways. The aggregate effect of these availability improvements can be quite dramatic when each individual feature is exploited in combination with the other availability enhancers.

Data: Fast Path DEDBs allow a database to be partitioned into multiple independent areas, and any or all areas can be replicated (up to seven copies).

Full function databases also have high availability characteristics in that, after write errors, IMS maintains the updated data in special buffers, and the data remains available (including across restarts). Database recovery can be deferred to the most convenient time.

IMS transaction programs are scheduled regardless of whether all databases are available, and the DL/1 API allows the program to manage the situation of unavailable data. Further, the API also enables BMP programs to recover and continue processing after deadlock failures.

Terminals: IMS detects and can automatically restart hung VTAM terminals. In IMS 4.1, the time before such a situation is detected has been reduced to just two minutes.

Rogue Programs: IMS provides protection against applications that take, but not release, thousands of locks on the database data. Similarly, it detects and terminates programs that attempt to flood the message queues.

MSC Links: IMS 4.1 provided significant new facilities for minimizing the impact of failed MSC links, and IMS 5.1 includes the ability to resolve “disagreements between IMS and VTAM” regarding MSC link status.

8.3 Security and Integrity

Data integrity for both databases and communication data is an IMS tradition. It distinguishes IMS from any other transaction and database manager environment on any platform. IMS was one of the first products to include a two-phase commit capability, with a syncpoint manager component.

8.3.1 Security

IMS security was first implemented many years ago and was done entirely by IMS, using SMU. With the advent of specific MVS security products, notably RACF, IMS has progressively been reducing the security functions for which SMU is still required.

RACF and equivalent products work at the user level. A user has to identify and validate his or her userid, and then RACF authorizes that userid against the many protected resources (for example, terminals, transactions, data sets) for which access is requested.

IMS 4.1 saw the introduction of RACF security for IMS commands. IMS 5.1 introduces RACF security for the new type of automated operator programs.

SMU works typically at the terminal level and is still available in some environments. But facilities like ETO are incompatible with SMU, and a move to user based security is recommended.

About the only facility left that requires SMU with IMS 5.1 is security for application group names (AGNs), which in fact is a facility provided jointly by SMU and RACF.

DL/1 AUTH Call

The DL/1 AUTH call can be used to request RACF checking of user access to any definable IMS resource (for example, database segment, keyrange, printer) and so enables very specific security controls to be imposed.

Message Security

As part of the ETO implementation, IMS provides facilities to help ensure that transaction replies are not seen by unauthorized personnel. It allows replies to be queued to the end user's userid (rather than to the terminal), so that messages for a user will be delivered only to the terminal where that user is logged on and signed on. Further it provides timeout facilities so that inactive users get automatically signed off the system.

8.3.2 Integrity

IMS integrity encompasses data integrity in database content and message integrity in terminal traffic. The use of the two-phase commit protocol and the inclusion of all resource managers capable of supporting a two-phase commit give IMS a substantive edge—one that will grow in the future as additional resource managers are added.

Data Integrity

It goes without saying that the IMS database manager provides first class data integrity. The two-phase commit protocol, used by IMS, ensures that only complete units of work (units of recovery) are applied to the databases. Even in a remote site recovery scenario, this is still true for remote databases with the IMS 5.1 RSR feature.

DBRC, used at the recommended level of Share Control, provides database protection from operational error and misuse. DBRC not only prevents wrong actions being taken but also makes it far easier to determine and execute the right action.

Message Integrity

Curiously, this major strength of IMS often goes unrecognized. Problems are almost nonexistent—IMS does not process the same transaction twice, replies from update transactions are assured of delivery to the terminal—and so the IMS benefits tend to pass unnoticed. However, this message integrity feature of IMS really should not be taken for granted. Other transaction processing systems tend to leave message integrity to the user, either by allowing user exits or by expecting that the applications themselves will include options to check the state of affairs after problem situations. In reality, many non-IMS customers run without message integrity and hope the end users will be able to recognize when problems occur and correct the situation!

The message integrity issue becomes especially hard to manage when executing in a distributed environment. For many years IMS has provided message sequence numbers for support of intelligent workstations defined as SLUP in IMS (LU0 protocol in SNA). This has allowed the partner systems to compare the send and receive sequence numbers after connection failures, and so avoid lost or duplicated messages. Intersystem communication or ISC (LU6.1 protocol in SNA) uses the same technique. In IMS 5.1, OTMA also supports sequence numbers, where appropriate, between IMS and the MVS client applications.

Message integrity is generally not provided with the more modern distributed processing models; it is left for applications to provide. The good news for users of the APPC/IMS implicit interface (that is, using traditional IMS applications to service LU6.2 devices) is that IMS still provides its standard integrity facilities because it has a recoverable log and has set standards for handling error situations. The remote LU6.2 programs must, of course, be written to be compatible with the IMS solution.

8.4 Protected Investment

Some customers today are running application programs written 15 or more years ago. The fact that they have no choice in continuing to run these programs unchanged, because the source code is lost, is not relevant. The key point is that, throughout the years, IMS has deliberately maintained a policy of application upward compatibility.

This does not mean that the DL/1 call API has remained unchanged. Indeed, it is now recommended that customers use a new form of DL/1 call when writing new applications. The call references a new control block, the AIB, rather than the traditional PCB, and this results in a much more powerful interface.

New function has been added, such as the INQY, SETO, SETU, and other calls. New status codes have been defined to enable specific situations to be reported back to the program rather than cause an abend. In Fast Path DEDB, new pointers (the SSP subset pointers) are available for applications to exploit. As mentioned earlier, BMPs can now recover from deadlocks. The JES spool interface allows alternate PCBs to be used to send messages to AFP printers with full two-phase commit support (if express PCBs are not used).

So the DL/1 API continues to get richer, but always while maintaining compatibility for previous releases.

The IMS developers recognize the time, skills, and money that have been invested in the application development efforts of IMS customers. Wherever possible, the traditional or legacy applications are being enabled to run in the newer open and distributed environments (without requiring changes to these existing applications). Since IMS 4.1, APPC/IMS has provided the implicit mode to allow LU6.2 programs and devices to execute traditional IMS transactions. In IMS 5.1, OTMA provides the equivalent facility for other networks and processing models.

Again, this does not imply that IMS is looking back. On the contrary, new applications can exploit new techniques (such as CPI-C or ATB calls, message queuing interface, TCP/IP sockets). These are covered in sections that follow.

8.5 The Open IMS Server

For many years, IMS has provided transaction serving for an SNA network and has supported most of the protocols defined in the SNA architecture, such as LU0, LU1, LU2, and LU6.1. IMS 4.1 included the APPC/IMS feature, which provides support for the LU6.2 protocol.

The new market need is for connection to any network, not just SNA (TCP/IP is the obvious example). The complementary need is to provide open access to IMS applications or data. IMS follows the guidelines proposed in both the MVS Open and Distributed Strategy and the IBM Networking Blueprint.

The following distributed programming models are emerging:

- **Conversational**, using the SAA CPI-C
- **Messaging**, using the messaging and queuing interface (MQI)
- **Remote Procedure Call**, using the RPC interface.

These models need to be supported across all network protocols (for example, SNA, TCP/IP, and OSI).

This section gives an overview of the IMS solutions for open and distributed environments and shows how IMS provides support for heterogeneous interoperability.

We consider the evolving role that IMS plays in this new world of transaction processing, as other related products continue to be developed and enhanced.

8.5.1 Conversational Processing with APPC/IMS

IMS 4.1 provides, in APPC/IMS, a full IMS implementation of the LU6.2 support, which permits program-to-program communication throughout the entire network. It uses APPC/MVS and APPC/VTAM services to provide the communications support.

Two possible types of application programs can run in IMS dependent regions:

- Traditional applications using IMS DL/1 calls to access messages (**implicit** mode)
- New applications using Common Programming Interface for Communications (CPI-C) calls to access messages (**explicit** mode).

The ***implicit*** API uses the IOPCB, as IMS applications have always used. The application program can access its data from DL/I or DB2 databases with integrity by using the standard DL/I calls. Communication with LU6.2 terminals is done by IMS on behalf of the application program, which continues to use the IOPCB and/or alternate PCBs to insert the data to be sent to the terminals.

The implicit interface offers two significant benefits:

- Traditional (legacy) applications, written for 3270s or SLUP devices, for example, can generally be used without change to process transactions from LU6.2 devices.
- IMS provides message integrity on the MVS system. All messages going through the message queues are logged and made recoverable where appropriate (especially output replies). The APPC conversational model assumes that applications take responsibility for message integrity. But for the IMS application programmer, this responsibility is provided by IMS and does not require application programming.

The ***explicit*** API uses CPI-C or ATB calls to communicate with the terminal.[10] The application program can still access IMS database and terminal resources, with IMS providing full commit point services for local resources (including DB2). IMS is not involved with data transfer over the LU6.2 conversation but does provide its customary services for program scheduling and data integrity.

IMS provides integrity and recovery capabilities for *IMS resources* in all cases, independently of the way the APPC partners are accessed or their particular capabilities.

Implementation Examples

In the figures that follow, you will find two examples of LU6.2 implementations using traditional IMS applications. In Figure 34 on page 154 the workstation is connected directly to IMS.

[10] MVS provides two APPC implementations, the CPI-C interface, which is common to many platforms, and an MVS-specific interface where all verbs are of the form “ATB.....” An IMS application can use either interface.

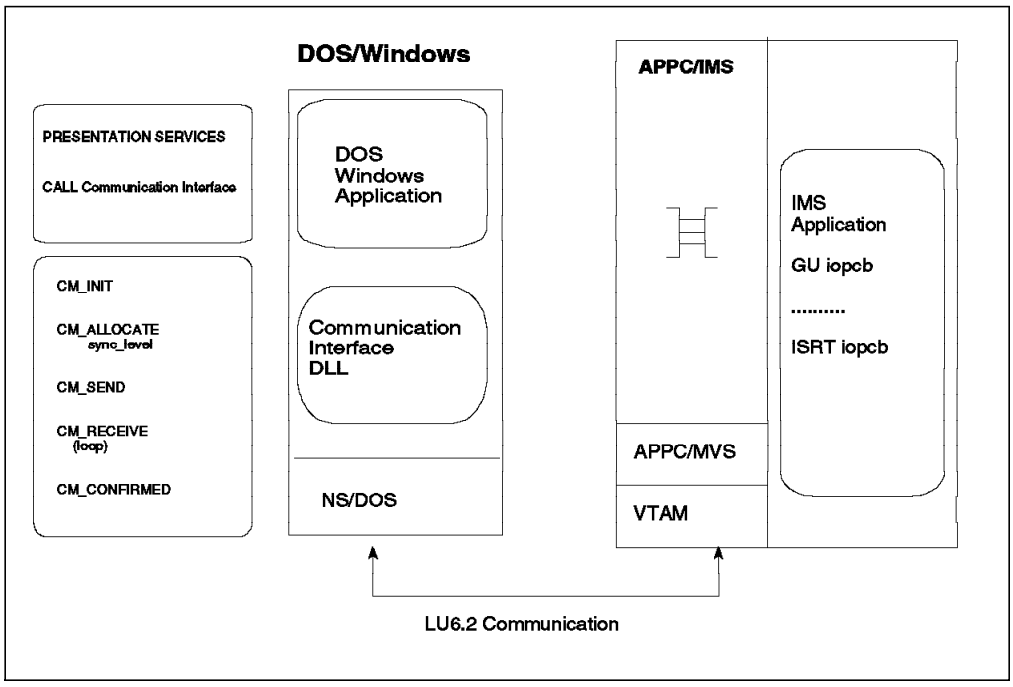


Figure 34. APPC/IMS: Example 1

In Figure 35 and Figure 36 on page 155) the workstation is connected to a transaction manager on a server (CICS OS/2 or CICS/6000), and the server handles the APPC communication with IMS.

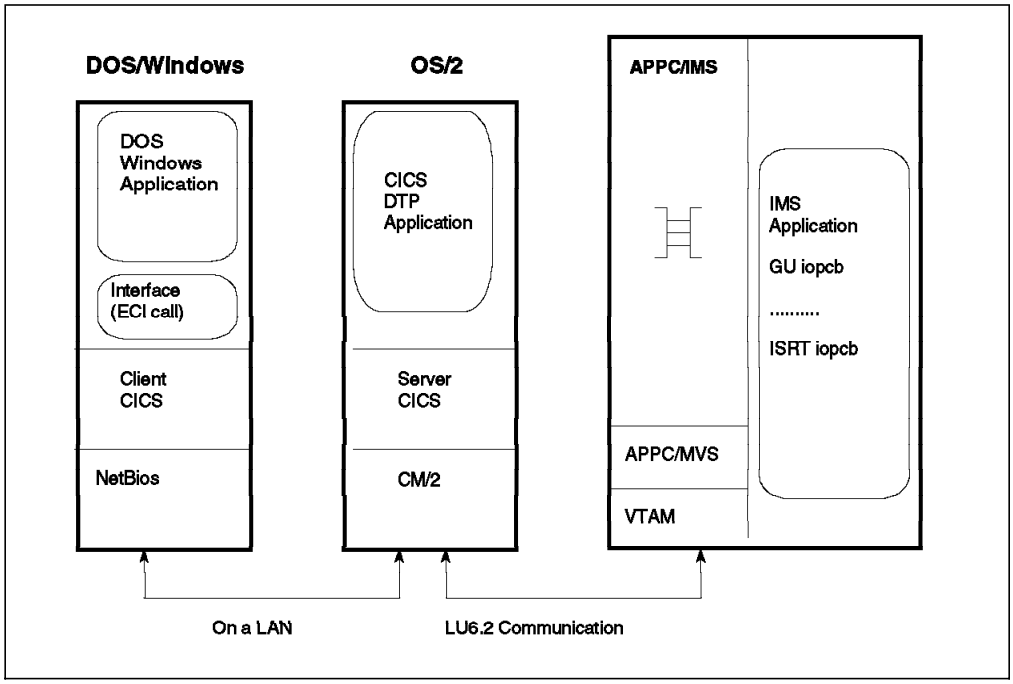


Figure 35. APPC/IMS: Example 2 - Architecture

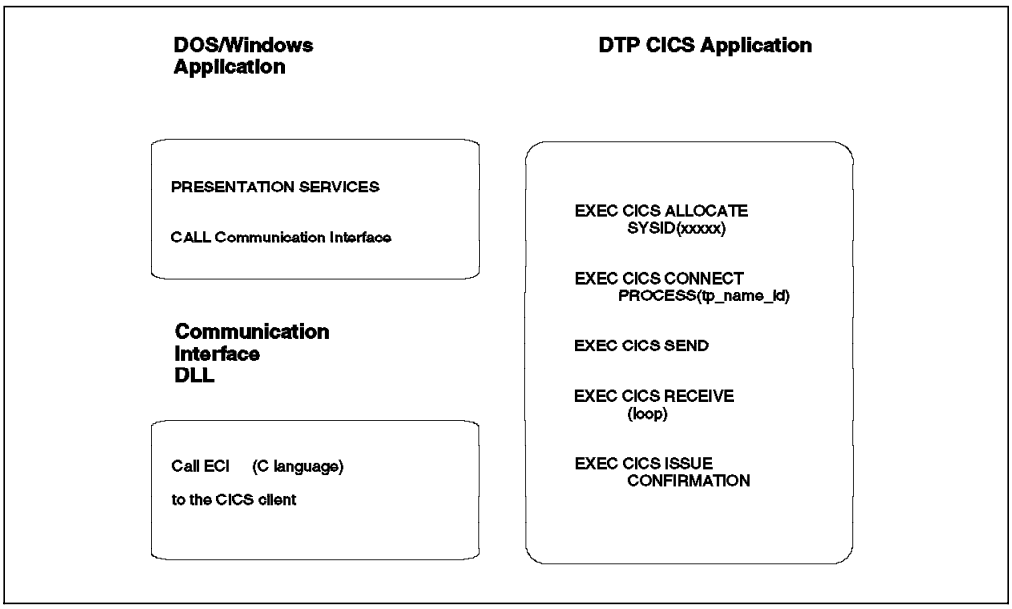


Figure 36. APPC/IMS: Example 2 - Programs

8.5.2 Messaging and Queuing

The messaging and queuing concept is very well known to IMS users because IMS has for many years provided a very powerful messaging and queue-based system with a high-level API.

Now the MQSeries family of products provides a common interface (MQI) for communications between applications in a heterogeneous network. IMS applications can already use the MQI with MQSeries for MVS/ESA (formerly MQM/ESA).

The distinctive benefits of the MQSeries family of products are:

- There is one interface across IBM or non-IBM platforms.
- Developers are shielded from networking complexities.
- Processing is time independent.
- Robust middleware is available for distributed applications.

The MQI is available in C, COBOL, PL/I, or Assembler.

MQSeries for MVS/ESA Environment

As well as providing connectivity to other MQSeries products, MQSeries for MVS/ESA offers in the MVS environment:

- Support for IMS, CICS, MVS Batch, and TSO applications
- Access using MVS cross memory services
- Participation in the transaction manager syncpoint processing (two-phase commit protocol)
- Support for persistent or nonpersistent messages.

Figure 37 on page 156 shows one example of how IMS applications can implement the MQI.

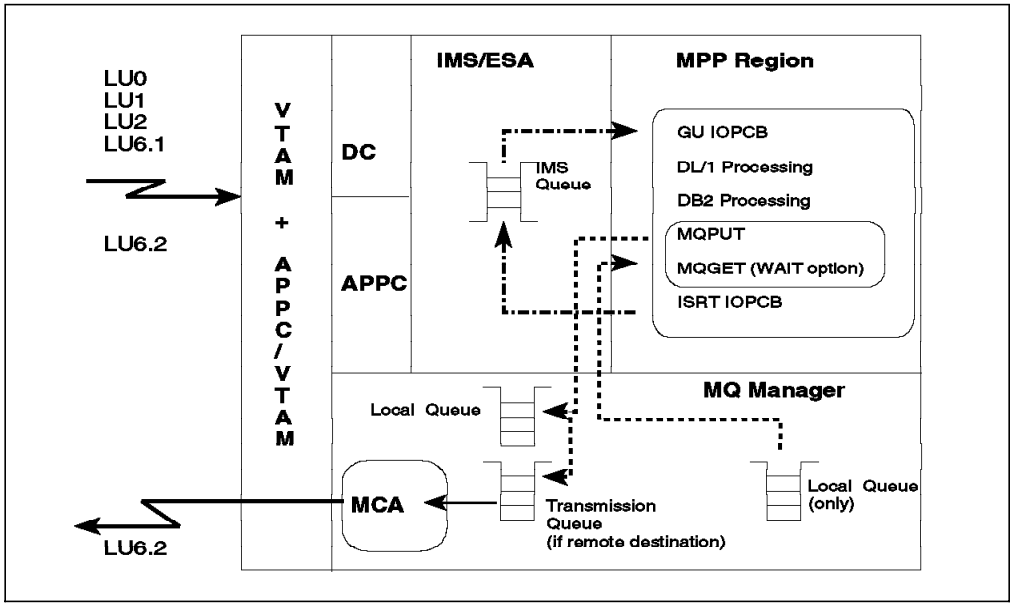


Figure 37. MQSeries for MVS/ESA with IMS

The message channel agent (MCA) is an MQ component that is used to move the messages between MQ managers through the entire MQ network. For the actual implementation on MVS systems, it uses the LU6.2 protocol.

MQM IMS Adapter in an IMS TM Environment

The interface for MQSeries for MVS/ESA implemented in IMS is called the IMS Adapter in the message queuing literature. It allows an IMS application to issue MQI calls to MQSeries for MVS/ESA and participate in the syncpoint processing of the IMS Transaction Manager.

The IMS Adapter uses the IMS External Subsystem Attach Facility (ESAF). The main components of the IMS Adapter, illustrated in Figure 38 on page 157, are:

- The API stub program, CSQQSTUB, which supports all verbs of the MQI
- The External Subsystem Module Table (ESMT), which specifies to IMS which modules from the External Subsystem Attachment Package (ESAP) are to be loaded in the IMS control region or the dependent regions when needed. Table 14 on page 157 lists the ESAP modules and their functions.
- The language interface module, DFSLI000, provided by IMS 4.1 and 5.1, already supports MQSeries for MVS/ESA. (The interface provided by the MQ manager, CSQQLI00, is not needed).
- The last component of the IMS Adapter is the IMS Trigger Monitor, which is described in “IMS Trigger Monitor” on page 158.

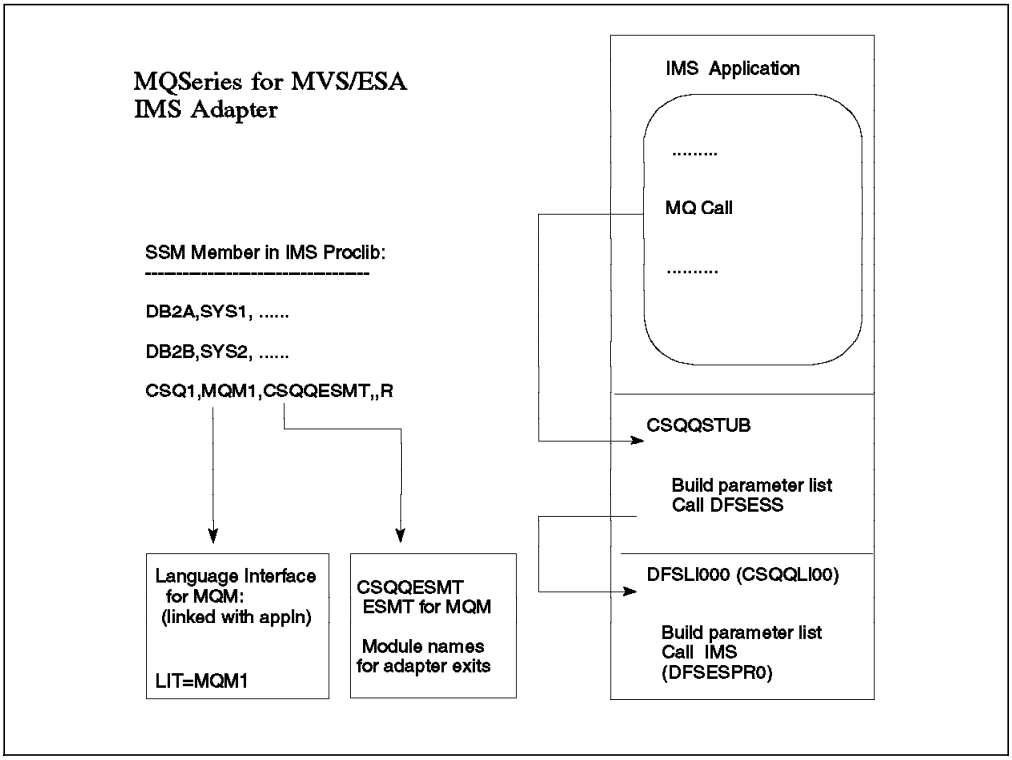


Figure 38. IMS Adapter: Connection Elements

Table 14. ESAP Modules for MQM IMS Adapter	
Module	Function
CSQQINIT	Initialize Exit
CSQQIDEN	Identify Exit
CSQQSSON	Signon Exit
CSQQCTHD	Create Thread Exit
CSQQPREP	Commit Prepare Exit
CSQQCMMT	Commit Continue Exit
CSQQBACK	Abort Continue Exit
CSQQTTHD	Terminate Thread Exit
CSQQSOFF	Signoff Exit
CSQQTMID	Terminate Identify Exit
CSQQSNOP	Subsystem Not Operational

The IMS application uses a language interface token (LIT) to identify the subsystem that is being called.

A PROCLIB member contains the subsystem names to be available for connection to the IMS control region or the dependent region. Its name is a concatenation of two parameters, the IMSID and the SSM specified in the startup parameters. The entry shown in Figure 38 contains:

- CSQ1 as the MVS subsystem identifier for the queue manager.
- MQM1 as LIT for the message queue manager called CSQ1.

The MQM language interface corresponding to that LIT must be link-edited with the IMS application. The LIT connects an application's subsystem call to a specific subsystem instance.

- CSQQESMT as the ESMT supplied by MQSeries for MVS/ESA. It contains the module names for the Adapter exits.

One IMS dependent region can be connected to more than one instance of MQSeries for MVS/ESA. Similarly, one instance of MQSeries for MVS/ESA can be connected to more than one IMS address space.

MQM IMS Adapter in IMS Batch Environment

The IMS Batch (DLIBATCH or DBBBATCH) environment does not use the IMS ESAF.

An IMS batch application can issue MQI calls (as well as DB2 SQL calls), but *no two-phase commit syncpoint coordination exists*. In case of a failure at application checkpoint time, there is no way of guaranteeing that DB2 updates, DL/1 updates, and MQ messages are consistent with each other—for example, one resource manager may have committed and the other two may have aborted. The only totally safe solution is to run batch work in BMP regions.

IMS Trigger Monitor

Triggering is the ability to start a transaction when a message arrives on a queue. Several types of trigger events exist. For further information, refer to the following manuals:

- *SC33-0806 MQM System Management Guide*
- *SC33-1212 MQM Application Programming Reference*.

On MQSeries for MVS/ESA, there is a trigger monitor for CICS (since MQM/ESA 1.1.1) and for IMS (since MQM/ESQ 1.1.2 - July '94).

Figure 39 on page 159 shows how the IMS trigger monitor works to start an *IMS modified application*. Figure 40 on page 159 shows a way of using the IMS trigger monitor to start IMS legacy applications without modifying them. This is obviously not the most ideal of solutions. It is possible, however, with a small degree of programming effort, to write the "transaction executive" shown in the middle of the Figure 40 on page 159. A much simpler solution for the future is described in "MQSeries for MVS/ESA to IMS Bridge" on page 160.

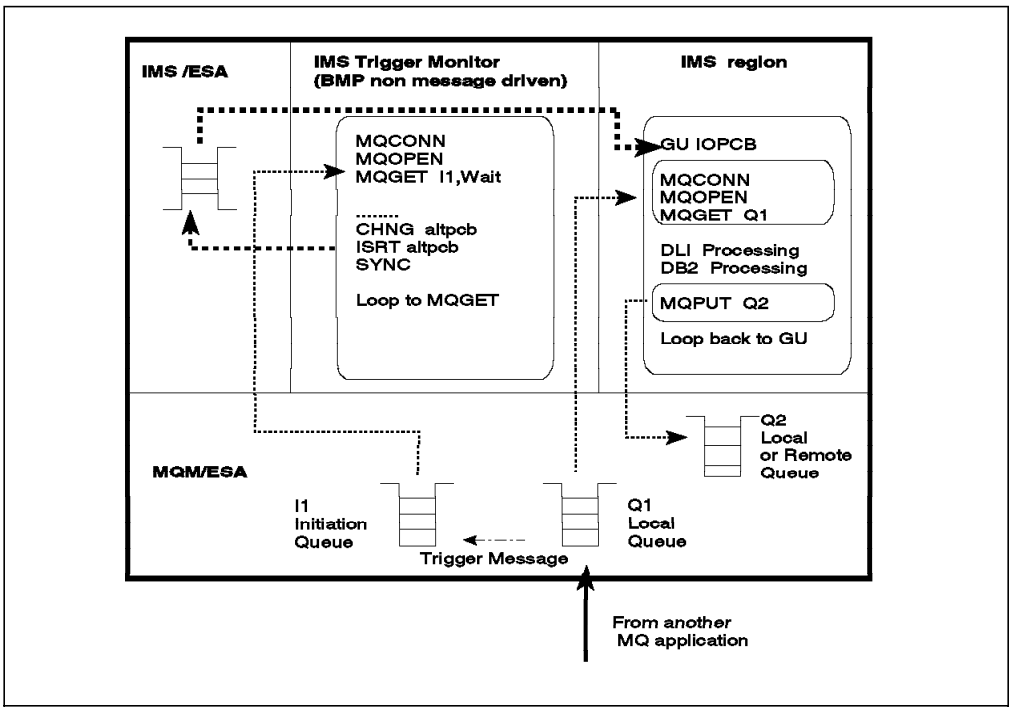


Figure 39. IMS Trigger Monitor: New IMS Applications

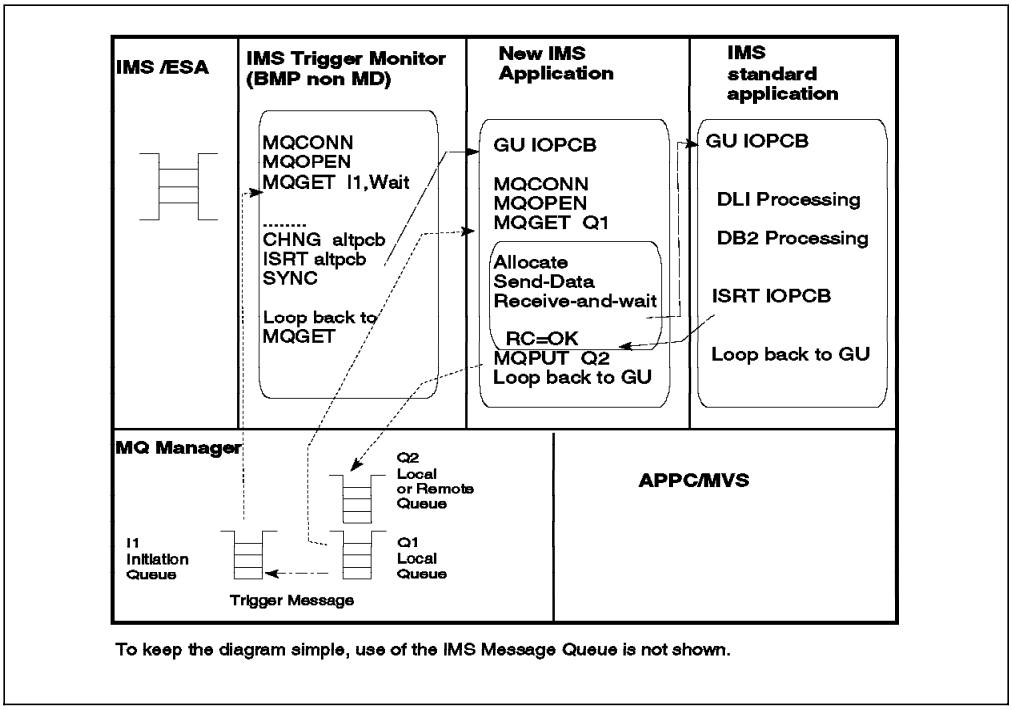


Figure 40. IMS Trigger Monitor: Legacy IMS Applications

The IMS trigger monitor is a long-running, non-message-driven BMP. A DD card, CSQQUT1, must be added in the BMP JCL. The card identifies to the trigger monitor:

- The queue manager to which to connect
- The initiation queue to open
- The IMS LTERM for error messages.

This BMP opens the initiation queue and waits for a trigger message to arrive. It wakes up periodically and checks for a trigger message on the initiation queue. It analyzes the trigger message and inserts a corresponding message into the IMS message queue. The IMS processing will be scheduled, and the message will be passed to the IMS MPP servicing this transaction. The IMS application is responsible for issuing the MQGET to the local queue. The name of this queue is specified in the **QName** field of the message received by the GU call to the IOPCB.

MQSeries for MVS/ESA to IMS Bridge

The MQSeries for MVS/ESA to IMS Bridge provides direct support for MQPUT to the IMS message queues. It uses the OTMA interface. The STGCLASS parameter must specify an XCF group name and an IMS member name, so that MQPUT will go directly to the IMS queue.

Figure 41 shows a traditional IMS application being invoked by a messaging and queuing input message.

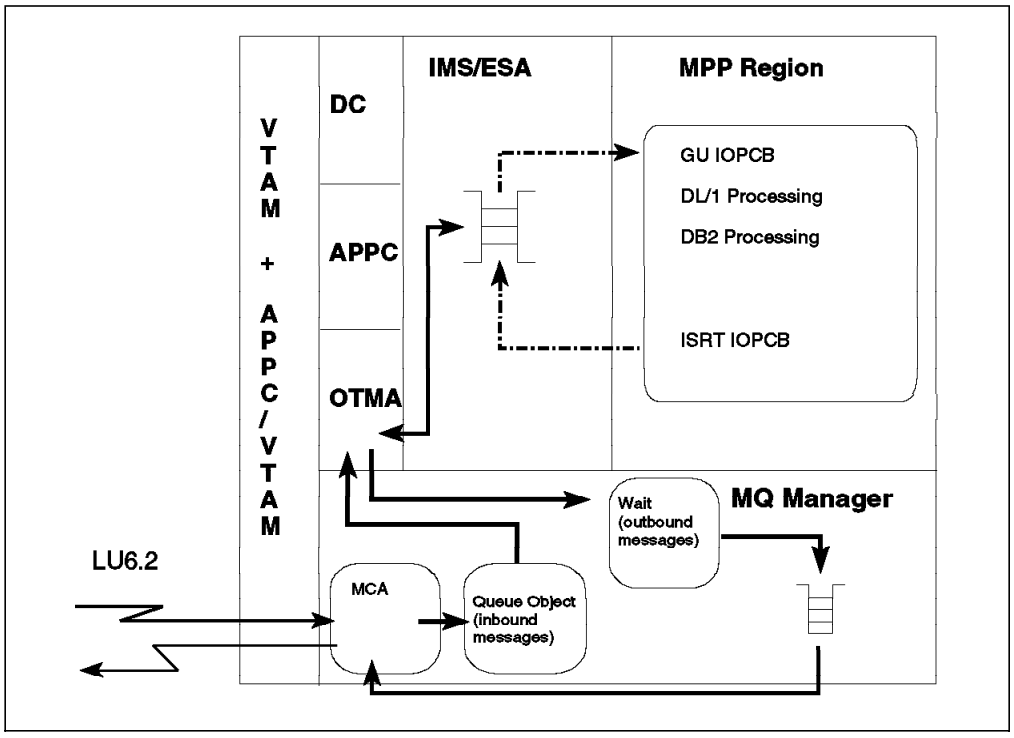


Figure 41. IMS Bridge

Without the IMS bridge, you cannot use existing IMS applications without modification or without writing an intermediate transaction executive program.

New applications can be designed to take benefit of messaging and queuing products. In an MVS environment, you can use MQSeries for MVS/ESA to communicate between a transaction application program and a batch application program.

8.5.3 TCP/IP Network Access

In a UNIX environment, IMS applications must be accessible through both SNA and TCP/IP networks.

Such accessibility is available through the IBM Networking Blueprint, the IBM Open Blueprint, and the AnyNet family of products. With ACF/VTAM V3 R4.2 for MVS/ESA with the AnyNet feature, LU6.2 APPC communication is supported on a TCP/IP network. The LU2 protocol is also supported on a TCP/IP network.

The next step will be the implementation of TCP/IP sockets, to allow network communication directly with IMS, without going through VTAM and APPC/IMS. Figure 42 shows the different routes to access IMS transactions from either SNA or TCP/IP networks.

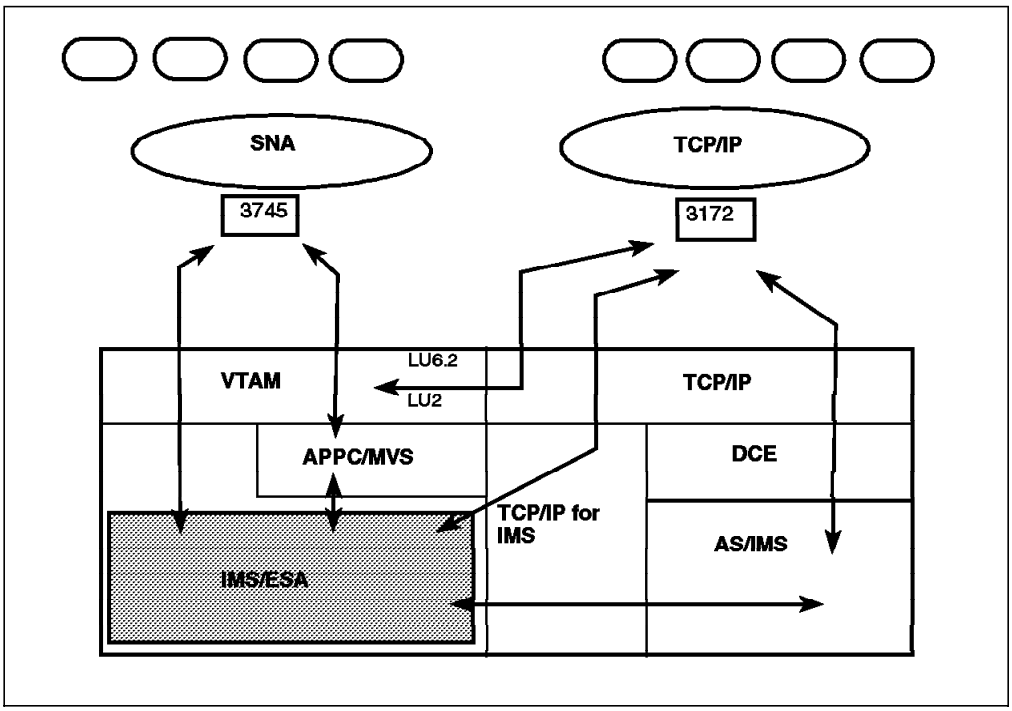


Figure 42. TCP/IP Access

TCP/IP for MVS

The TCP/IP for MVS product provides TCP/IP services in the MVS environment:

- The TCP/IP engine
- The socket API

  This API provides access to UNIX services but only for application programs written in C.
- A set of standard applications including file transfer protocol (FTP), simple mail transfer protocol (SMTP), and Telnet.
- CICS support
- **IMS support**

  This IMS TCP/IP feature simplifies the coding of TCP/IP-based IMS applications by providing:

- Sockets Extended API for COBOL, PL/1, and Assembler (without requiring C language)
- IMS Listener
- IMS Assist Module.

It is designed for use on an MVS host running IMS/ESA V3 (or later) and TCP/IP V3 R1 for MVS. See “TCP/IP for IMS” for more details.

LU2 over TCP/IP Network

Traditionally, many IMS users have used the 3270 protocol to communicate with IMS TM.

When a TCP/IP host needs access to a traditional 3270 MFS application, it can use the 3270 emulator function called Telnet (TN3270), a part of the base TCP/IP for MVS product, which provides 3270 emulation services for TCP/IP-connected clients.

APPC over TCP/IP Network

Applications written with SNA application interfaces must be able to go across TCP/IP networks. The initial solution is restricted to the LU6.2 protocol. Other solutions will come later.

This means that APPC applications which were primarily restricted to SNA networks are now available to end users in TCP/IP networks. Any workstation on a TCP/IP network can easily execute IMS transactions or CPI-C driven applications and so access IMS and DB2 databases.

Because the transport enveloping is an implementation at the network product level, no changes in APPC applications (or the IMS system using APPC) are required. On the MVS host, VTAM V3 R4.2 and the AnyNet feature are required as well as TCP/IP for MVS. The minimum level of MVS is MVS V3.1.3

TCP/IP for IMS

The TCP/IP for IMS feature of the TCP/IP for MVS product provides two-way client/server communication between an IMS MPP and TCP/IP-connected clients. With this feature, applications written with the TCP/IP sockets interface can access IMS transactions, and IMS client/server applications can be developed using the TCP/IP sockets API.

The components of this feature are as follows:

The IMS Listener: This component is the single point of contact for TCP/IP clients. It is a BMP program that waits for connection requests from TCP/IP-connected hosts. When a request arrives, the IMS Listener schedules the appropriate transaction and passes a TCP/IP socket, representing the connection, to that server. The IMS Listener holds the connection until the MPP starts and accepts the connection. Then the Listener is no longer involved.

The IMS Assist Module: This is an optional subroutine included in the IMS application program. It allows the use of DL/1 calls for TCP/IP communication between the client and the IMS application. Therefore IMS application programmers do not need to have TCP/IP skills.

- Implicit mode transactions

The application programs that use DL/1 calls are called **implicit mode** programs. Those transaction programs obtain their input message from the IMS message queue. They are limited to one input of one or more segments, followed by one output of one or more segments.

- Explicit mode transactions

  Application programs that use socket calls are called **explicit mode** programs. Those transactions bypass the IMS message queue for input and output messages.

The MVS TCP/IP Socket API (SOKETS without C): This is a new API provided for the TCP/IP for IMS feature.

The MVS TCP/IP socket call interface allows programs written in COBOL, PL/I, and Assembler to issue socket calls.

The IMS socket calls are a subset of the TCP/IP for MVS socket calls and are designed to be used in application programs written in languages other than C. Hence the term, **SOKETS** without C!

TCP/IP for IMS Message Flow

The figures that follow illustrate the message flow between the different components. For a detailed description, see the TCP/IP for IMS documentation.

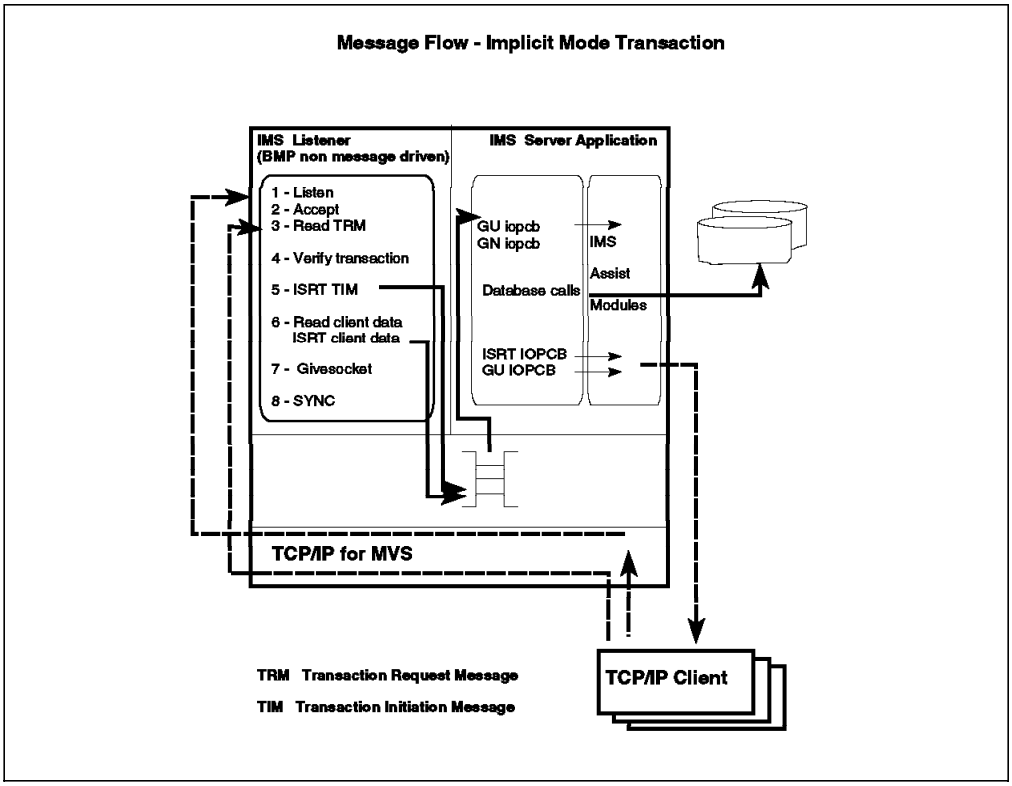


Figure 43. Message Flow for Implicit Mode Transactions

Referring to Figure 43 or Figure 44 on page 165, the steps for transaction initiation are:

1. Connection request
2. Connection processing

3. Receive Transaction-Request Message (TRM) containing the IMS transaction code.
4. Transaction verification
5. Transaction Initiation Message (TIM) put onto the IMS message queue (first segment of multisegment message)
6. This step does not exist for explicit mode transactions.

   For implicit mode, the IMS Listener reads the client-to-server input data and places it on the IMS message queue as the second segment following the TIM.
7. Pass the socket to the server
8. Syncpoint processing.

Referring specifically to Figure 43 on page 163, for implicit mode transactions, the data exchanges between the IMS MPP and the TCP/IP client are as follows:

1. GU to IOPCB must be the first IMS call. The IMS assist module intercepts the message and analyzes it to ascertain whether it is a TIM or a normal IMS message. If it is not a TIM, the subsequent calls are passed directly through to IMS. If it is a TIM (*LISTNR* in the first field), the IMS assist module establishes the connection between the client and the server program.

   Once the IMS server owns the socket, the IMS assist module issues a GN call to retrieve the client data and gives it to the IMS program in response to its GU IOPCB.
2. The IMS assist module accumulates the output segments resulting from ISRTs to the IOPCB.
3. A GU IOPCB implies syncpoint and transaction completion. The IMS assist module then sends the data to the client and asks the Transaction Manager to commit databases changes.
4. If the IMS syncpoint is successful, the IMS assist module sends a complete status message segment (CSM) to the client, closes the socket, and ends the connection.
5. The program can loop on the GU IOPCB, as usual.

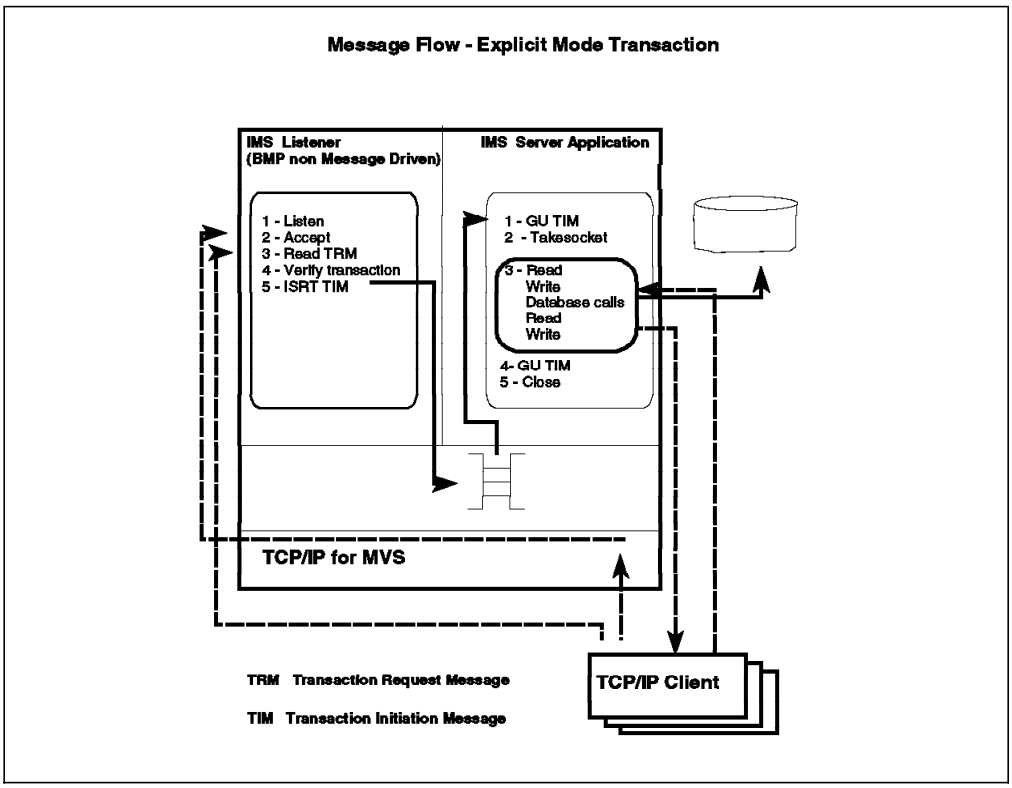


Figure 44. Message Flow for Explicit Mode Transactions

Referring specifically to Figure 44, for explicit mode transactions, the data exchanges between the IMS MPP and the TCP/IP client are as follows:

1. Once an explicit mode server begins execution, it issues a GU to obtain the TIM.
2. Then it issues a TAKESOCKET call to establish direct connection between the IMS server MPP and the client.
3. Socket READ/WRITE commands are used to exchange data between client and server.
4. Updates to databases can occur.
5. A GU IOPCB implies syncpoint and transaction completion. No messages are sent to the client through the IMS message queue.
6. The program can loop on the GU call.

8.5.4 DCE/RPC Support through AS/IMS

The remote procedure call allows applications to access procedures on a network as if they were local subroutines.

IMS supports the Open Software Foundation's Distributed Computing Environment (OSF/DCE) RPC model, in cooperation with MVS OpenEdition and OpenEdition Application Server/IMS (AS/IMS).

Figure 45 on page 166 to see how the DCE/RPC support is implemented to give access to existing IMS applications. One objective of DCE/RPC is to simplify the development of distributed applications by removing the overhead of writing communications code. Another objective is to allow existing applications to be distributed with minimal change.

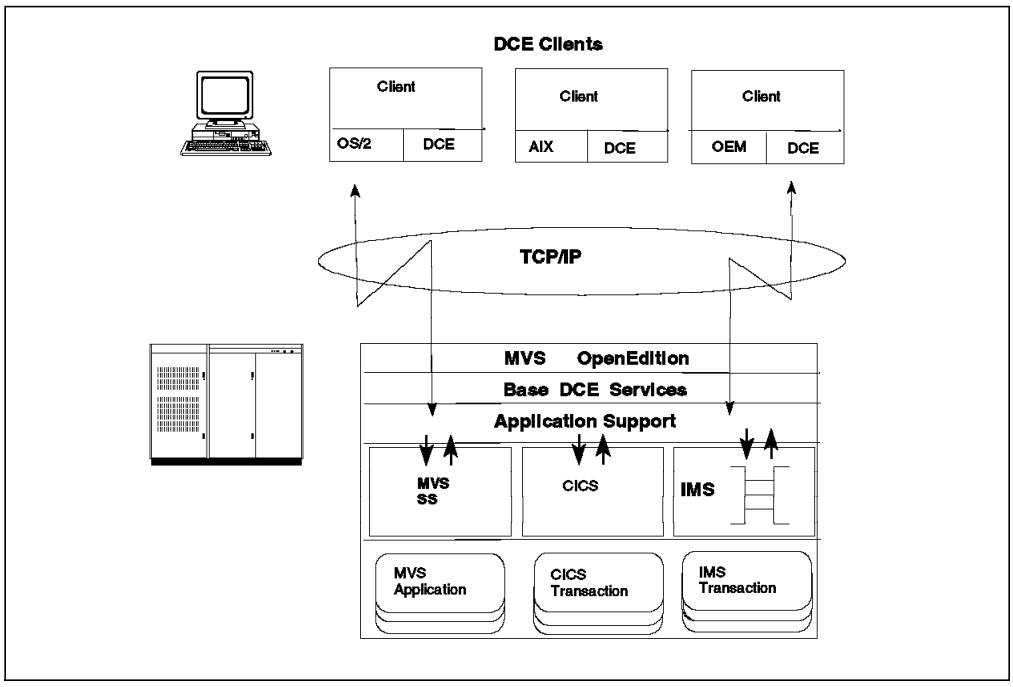


Figure 45. DCE/RPC Implementation

The IMS Transaction Manager can participate as a server in a DCE complex. Any open platform supporting DCE can initiate IMS or CICS transactions and can access IMS or DB2 databases. AS/IMS and AS/CICS are the interfaces between the Transaction Manager and the DCE environment and form a bridge between the new heterogeneous world of DCE and the traditional world of MVS. The application servers thus not only protect the existing investment but also minimize connection costs.

RPC Client

On the client side, the components are base DCE components that can interoperate with other client/server components in the DCE environment.

The DCE client applications, written in C, issue RPC calls to AS/IMS to invoke existing IMS COBOL applications. All types of IMS applications are supported. Because the front-end application involves only a DCE RPC program, it can easily be ported to any other DCE environment.

IMS RPC Server

AS/IMS receives the request from the network and, using an ISC (LU6.1) link, submits the IMS transaction to be executed. The output message, inserted on the IOPCB, is sent back to the client application by AS/IMS.

AS/IMS converts COBOL and C data types automatically. It may support transaction programs written in other than COBOL if the input and output parameters are equivalent to the COBOL data types.

IMS Restrictions When Using AS/IMS

- Only COBOL application programs
- Single segment input message (no restriction for output messages).
- MFS is not used. The formatting for terminal input and output must be in the client program.
- No IMS command support (it may work but is not officially supported)
- No message switching support
- IMS application programs must not have an LTERM name dependency.

8.5.5 Remote Procedure Call with the IMS Client Server Products

The IMS Client Server family of products, **IMS CS/2** and **IMS CS for Windows**, provides an implementation of RPC. There is a simple callable programming interface for workstation applications. It is used to call existing IMS applications that are accessible using the SNA LU2 protocol and use MFS 3270 screens. The objectives of these products are to:

- Protect and build on the existing investment in IMS 3270 applications
- Bridge host data into the OS/2 or Windows environments
- Provide application programmers with an easy-to-use API that will help in creating client/server solutions
- Enable customers to benefit from the power of the workstation (GUI interface, for example) and LAN connections
- Facilitate use of state-of-the-art programming tools to deliver new functions quickly and efficiently.

Workstation applications can provide services such as:

- GUI
- Integration of stored audio data
- Drag-and-drop utilization
- Execution of multiple IMS transactions to service one user request
- Online help.

Figure 46 on page 168 illustrates the IMS Client Server products.

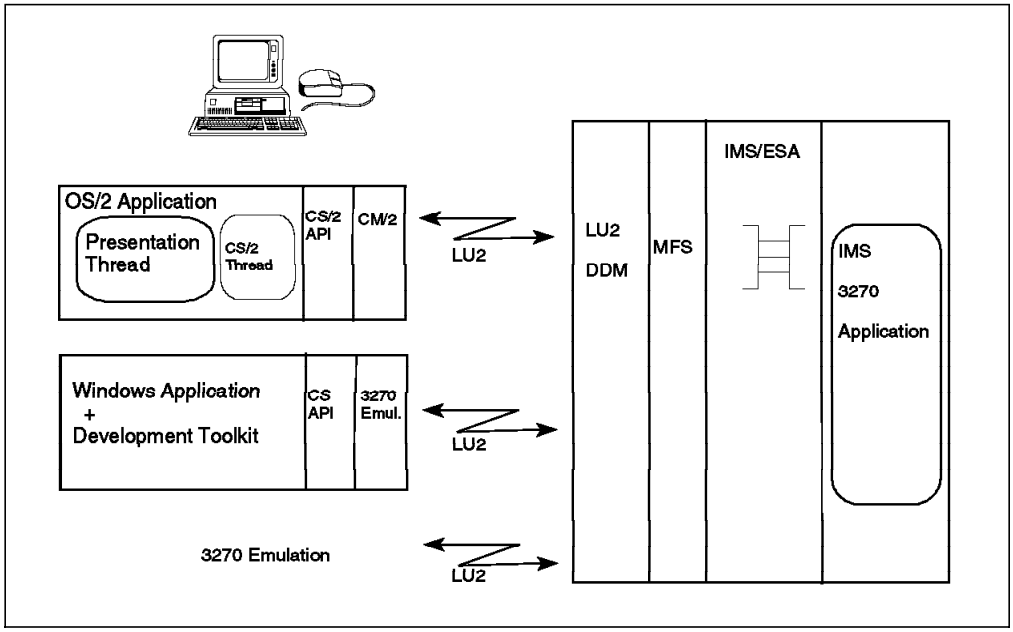


Figure 46. IMS Client Server Products

IMS Client Server/2

This workstation product runs on OS/2 workstations. It contains several features, some of which are needed only by the workstation application developer.

The end-user environment only needs the IMS CS/2 Runtime Base feature.

The developer environment also includes the Administrative and Development feature, which consists of:

- Dialogs to define the IMS configuration
- The MFS Conditioning Utility (MCU), which adds a unique identification to the IMS format (the IMS CS/2 literal) and creates a view of these formats in the workstation environment.
- An online test facility and online help.

IMS CS/2 Version 2 Release 2 supports:

- VisualAge, IBM′s new object-oriented, visual application development environment.
- Workstation applications written in PL/I, COBOL, C, FORTRAN, or REXX.

For more information, see the following IMS CS/2 documentation:

- *SC26-3256 IMS CS/2 User′s Guide and Reference*
- *GG24-3812 Cooperative Processing Solutions Using IMS CS/2*.

IMS CS for Windows

IMS CS for Windows is a set of runtime Windows DLLs that provides the communication links between IMS 3270 applications and the Windows environment. It can be used with any Windows development tool that is capable of calling DLLs. One such tool is the IMS Client Server Toolkit (Multi Soft Corporation) that provides a high-productivity development environment. It works in conjunction with the IMS CS/2 Conditioning Utility.

For more information, see the following IMS CS for Windows documentation:

- *SC26-3417 IMS CS for Windows User's Guide*
- *SC26-3418 IMS CS for Windows Reference Manual*.

8.5.6 IMS Client Server Object Manager

The IMS Client Server Object Manager (IMS CSOM) for OS/2, middleware that enables object-oriented client access to enterprise data, combines the benefits of the workstation as a development environment with access to critical data at the mainframe. IMS CSOM middleware consists of three tiers: a client tier, a server tier, and a datastore tier. It enables an OS/2 client to access and update both IMS and DB2 databases using an object-oriented API.

Client application programs use the common interface of the Object Management Group's Persistent Object Service to manipulate objects. This interface features a set of classes and methods that the client application program uses and builds on. Client application programmers can develop applications with little or no knowledge of the host environment.

On the workstation server tier, IMS CSOM maintains and manages a recoverable database of persistent objects that can be accessed and changed by clients of the server. IMS CSOM requests data objects from the IMS hosts and routes the objects to the server as requested.

The benefits of using IMS CSOM are:

- Access to IMS data through an object-oriented, insdustry standard API
- Ability to develop workstation applications using object-oriented techniques and software while retaining a safe, secure, and reliable data store
- Control of data access performance by caching frequently used data in the server tier near the workstation
- Flexibility in future software usage.

8.6 IMS Application Portability

For IMS, portability does not mean that an IMS application will run without any modification under another operating system. IMS needs and will always need MVS. The main direction is to give access to IMS applications from heterogeneous networks and provide application *programmer* portability by allowing application programs to use standard protocols:

- Common call API, which is DCE/RPC

  By using Application Server/IMS
- Common database API, which is SQL

  By using DB2

  By using non-IBM products that have implemented SQL on DL/1 databases
- Common language support, which is C

  Since IMS 4.1
- Common network protocol support, which is TCP/IP

  By using TCP/IP for MVS

- Common systems services interface, which is POSIX

  By using MVS OpenEdition.

Chapter 9. Migration to IMS 5.1

The migration process consists of many steps. Some of these steps are mandatory; others are optional. The optimal pace and scope of migration will vary for each customer environment.

9.1 General Approach

A wide variety of customers use IMS systems for many different applications and in many different environments. The actual migration path and implementation schedule that are best for each customer are highly installation dependent.

As a general approach, if the system is important, preserving fallback capability is essential. The migration process should be structured to maintain this fallback capability. We recommend the following migration scenario:

1. Select mandatory changes.
2. Select as many optional changes as possible for inclusion in the initial migration. This increases the migration effort and risk to some extent, but this increase is usually a small percentage of the migration effort. It is generally better to make changes now rather than deferring their implementation until after the version migration for the following reasons:
   - Preparation for the change can be done using the migration resources.
   - More thorough testing is possible because it is done in conjunction with the migration.
   - The impact of changes on end users can be minimized. End users are already impacted by the migration and need not be impacted later by additional changes.
   - The fallback procedures for the change provide coverage for the other changes.
   - There is greater stability after the migration because additional changes in software or procedures will not be necessary.
   - Special coverage before, during, and immediately after the migration period can be arranged by forming a “SWAT team” for quickly resolving problems as they occur.
3. Assess the impact of migration and plan fallback scenarios.
4. Evaluate testing requirement to ensure that it is representative of your system′s actual workload and criticality.
5. Perform regression tests and stress tests thoroughly.
6. Phase into production system.

Migration planning allows you to migrate safely and avoid an unexpected fallback. The migration plan should itemize what will change in this migration, what should be tested, who is responsible for completing the testing, which risks exist, and the fallback process for handling each risk.

The scope and complexity of the plan should reflect the importance of the IMS system being migrated.

9.2 Migration Paths

Only two migration paths to IMS 5.1 are supported:

- IMS 3.1 to IMS 5.1
- IMS 4.1 to IMS 5.1

Migration to IMS 5.1 from other IMS versions is gated by compatibility of IMS logs, RECONs, and MSC links. Fallback to a previous IMS version is difficult or impossible outside of the supported migration path.

9.3 Discontinued Support

Two facilities have been discontinued in IMS 5.1:

- Local DL/1 for CICS

  All users of CICS Local DL/1 must convert to DBCTL to use IMS 5.1. Discussion of the ways of achieving conversion can be found in *IMS/ESA 4.1 Migration Guide, GG24-4150*.

- LU6.2 adapter

  There is no support for the adapter, although the code is still shipped with IMS 5.1. Users of the adapter who do not need MSC support for LU 6.2 should convert to using native APPC/IMS support before migrating to IMS 5.1. If MSC is needed for LU6.2 devices, the conversion from the adapter to APPC/IMS can be deferred as necessary. More information can be found in 7.6.2, “LU6.1 Adapter for LU6.2 Applications” on page 143.

9.4 Mandatory Migration Items

Migrating DBRC to IMS 5.1 is exactly as in previous IMS migrations. The basic steps are:

- Install and thoroughly test DBRC migration maintenance on current IMS systems (IMS 3.1 or 4.1).
- Back up existing RECONs using the current system.
- Install IMS 5.1.
- Execute the IMS 5.1 RECON Upgrade Utility.

  Should this fail, recover RECONs from the backup copy before trying again.
- Delete and redefine discarded RECONs.

Almost all of the RECON records have changed, but, as in IMS 4.1, mapping macros are shipped with the IMS code.

9.4.1 COMM Macro

The AOEXIT parameter must be removed.

The MSTEXIT or NOMSTEX parameters should be removed.

9.4.2 VSAM Database RACF Checking

Ensure that IMS Control Region and DLISAS have RACF userids assigned that are authorized to access all of the online database data sets (see 7.1, “Enhanced Database Access Security” on page 139).

9.4.3 Fast Path

The migration process should evaluate the usage of utilities and operating procedures in the areas discussed below. The changes are minor and easily accommodated for most systems.

IOVF Statistics on /DIS AREA Command

The /DIS AREA command now requires an additional parameter if the count of available IOVF CIs is to be displayed (see 7.4, “Change to /DISPLAY AREA Command” on page 142).

HSSP Backward References

HSSP application programs can no longer make backward references using the HSSP PCB (see “Migration Implications” on page 40).

HSSP Image Copy

The HSSP Image Copy no longer exists. It has been replaced by the HSSP Asynchronous Image Copy (ASIC); see 3.2.13, “HSSP Asynchronous Image Copy” on page 41.

HSSP Storage Requirements

HSSP gains its performance benefits in IMS 5.1 by exploiting larger amounts of data in private buffers. Consideration should be given to the real storage requirements of concurrent HSSP jobs (see “Migration Implications” on page 40).

Online Reorganization Utility Buffers

The BUFNO parameter has a new meaning and, as for HSSP, more real storage may need to be “allocated” to multiple concurrent uses of this utility (see “Implementation” on page 37).

Fast Path Log Tape Analysis Utility Changes

Any user-written programs that process output from the Log Tape Analysis utility will need to be modified.

9.4.4 IMS User Exits

Some minor changes in the calling environment on some user exits must be considered during the IMS 5.1 migration process. The following user exit changes need to be evaluated:

- DFSLULU0 (LU6.2 Destination Exit)

  No longer exists. The function has moved to DFSCMUX0.

- DFSCMUX0 (Message Control/Error Exit)

  Now covers MSC, LU6.2, and OTMA.

  /DEQ commands are always enabled—the exit is no longer needed for this purpose.

- DFSCMPR0 (MSC Program Routing Exit)

Requires modification because it is now also called for local transactions.

Exits No Longer Linked with IMS Nucleus

- DFSCMTR0 (MSC Terminal Routing Exit)
- DFSAOUE0 (Automated Operator Exit - Type 1).

9.4.5 Log Records

Most log records have changed. Any user programs that process IMS log records must at least be reassembled with the IMS 5.1 DSECTs.

9.4.6 Message Queues

The Message Requeuer (product number 5665-038) is very useful for moving messages between IMS 3.1, 4.1, and 5.1 systems. We highly recommend that you install this product even if the need to move messages between IMS systems is not anticipated. Two important functions are enabled when the message requeuer is available:

- Ability to move messages to IMS 5.1 during migration and from IMS 5.1 during fallback
- Ability to capture and execute test data for regression and migration testing.

This capability to move messages between systems using the Message Requeuer can significantly ease the migration effort and is particularly useful in reducing the risk of workload loss if a fallback from IMS 5.1 must be executed.

APAR PN65138 for IMS 5.1 must be installed to support the Message Requeuer. This support allows migration of messages originated in an IMS 4.1 or an IMS 3.1 system to IMS 5.1. In a fallback situation it allows IMS 5.1-originated messages to be loaded on the message queue of an IMS 3.1 or IMS 4.1 system.

9.4.7 Implementing MSC with APPC/IMS

When MSC is used for message routing and the originating terminal is an LU6.2 device, *every system involved in the processing* must be an IMS 5.1 system. This includes the originating system, any intermediate systems, and the remote systems.

9.5 Optional Migration Items

The optional migration items are of lower priority because they generally do not affect application or system function. Although they require less attention, an assessment of their impact for the specific IMS 5.1 migration should still be made as in some special cases the impact may be undesirable. This analysis can usually be done in a few minutes so it is not a significant part of the migration effort.

9.5.1 Items without Fallback Inhibitors

The following IMS 5.1 features do not affect migration as they have either been retrofitted to IMS 4.1 or can be left in their IMS 5.1 state without any special processing for fallback (the IMS 5.1 level function will generally not be available but the system will operate correctly):

- DEDB PROCOPT=GO
- DEDB VSO (but not converted MSDBs)

- SVC Utility
- APPC/IMS LTERM and MODname support (retrofitted to IMS 4.1).

9.5.2 Items with Fallback Implications

The following IMS 5.1 functions do not have a direct counterpart in earlier IMS releases. In case of fallback the prior IMS system must be restored and able to operate as it did before the IMS 5.1 conversion:

- MSDB to DEDB (probably VSO) conversion
- MVS Work Load Manager
- /PSTOP LINK FORCE for hung MSC links.

Chapter 10. IMS 5.1 and DBCTL

This chapter briefly describes the advantages of the DBCTL environment compared with CICS Local DL/1 and summarizes what is new in IMS 5.1 for the DBCTL customer.

10.1 DBCTL Environment Summary

Before IMS 5.1, CICS users had the choice of two methods of accessing DL/1 databases from the CICS environment.

Now in IMS 5.1, the CICS Local DL/1 support has been removed. Migration from Local DL/1 to DBCTL must be done while still on IMS 3.1 or IMS 4.1.

What follows is a summary of the main advantages of the DBCTL architecture. Moving to DBCTL from Local DL/1 requires a significant culture change. But the conversion effort is well rewarded on entering the world of IMS system managed databases!

10.1.1 Separate Data Server

Implementing separate address spaces to handle the IMS work significantly reduces the storage requirements in the CICS address space by moving a significant number of control blocks and modules into the IMS address spaces.

Failure isolation also enhances the availability characteristics of the system. IMS and CICS abend and recovery processing are now separated.

The architecture of DBCTL is very similar to that of DB2, and this helps the CICS operator even though the DL/1 and DB2 cultures are rather different. Some of the similarities are:

- One system address space and one database address space
- Centralized log management for all the database updates
- Participation in two-phase commit.

10.1.2 Alternative to Multiregion Operation

DBCTL is the preferred alternative to multiregion operation (MRO) for the sharing of DL/1 databases between application programs in different CICS/ESA regions. It also provides a better alternative to CICS shared databases.

In a single MVS environment with only one DBCTL, DBCTL is the owner of the DL/1 databases and controls the access by using IMS PI locking services or the IRLM.

In a sysplex environment, DBCTL will be seen as an IMS subsystem concurrently sharing databases with other IMS subsystems (including IMS Batch) by using BLDS.

10.1.3 Centralized Log Management

The components of IMS logging are:

- Online log data set (OLDS)

  IMS writes log records primarily to allow recovery of databases, message queues, and the IMS system itself. All log records get written to the OLDS.

  Multiple OLDS are allocated and used in rotation. As each is filled, its log data is archived to make that OLDS available for reuse.

- Write ahead data set (WADS)

  The WADS is used to satisfy log write-ahead requests, essential for ensuring data integrity. The WADS is used to store partial blocks of log data until a log buffer is full and ready to be written to the OLDS. Writes to the WADS use a special I/O technique that provides exceptionally short I/O times.

  Only full size blocks (usually half a track) are written to the OLDS, thus providing optimum space utilization and performance.

- Secondary log data set (SLDS)

  Each OLDS archive creates an SLDS. Typically, SLDSs are held on tape.

- Recovery log data set (RLDS)

  The RLDS is an optional data set, produced at OLDS archive, and contains only the log records required for database recovery.

- Restart data set (RDS)

  The RDS is used to record system checkpoint information for use during IMS restart.

- DBRC RECON data set

  The RECON is required to record information about the logging subsystem.

  The role of DBRC in this respect is:

  – Managing and tracking the use of the OLDS
  – Controlling the Log Archive Utility (DFSUARC0)
  – Controlling the Log Recovery Utility (DFSULTR0).

10.1.4 Opportunity to Use DBRC

With CICS Local DL/1, very few customers use DBRC to protect their databases, because its implementation requires approximately 900KB of virtual storage below 16MB in the CICS address space.

In the DBCTL environment, DBRC is already mandatory for IMS log control. It is recommended that DBRC also be used to control the databases themselves. DBRC offers a choice of two levels of database control:

1. Database recovery control

   DBRC tracks when databases were opened for update and which logs (SLDS or RLDS) were involved. It tracks database image copies and log accumulates. When a database needs recovering, DBRC can generate and run the recovery job. It improves operational productivity and helps minimize errors.

2. Data sharing control

   This is the preferred level of DBRC control.

As well as providing all recovery control functions, data sharing control tracks the status of databases and ensures that no invalid use is made of them. For example, it prevents access by online systems to a database requiring backout.

In an environment where multiple IMS subsystems share the same databases, DBRC will control the concurrent access to the databases. Databases are registered with DBRC with a specification of the sharing level:

- No sharing

  One subsystem at a time has exclusive use of the database.

- Database level sharing:

  Multiple subsystems can all have concurrent read access. Alternatively, DBRC will authorize one updating subsystem and multiple read-only (without integrity) subsystems.

- BLDS

  Multiple subsystems can have update or read access with full integrity.

  DBRC and IRLM are both mandatory.

10.1.5 Access to DEDBs

The benefits of DEDBs are:

- Very high performance
- Very high availability
- Support for very large databases.

These benefits are available to CICS DBCTL users.

With IMS 5.1, CICS can also make use of the in-memory facility for DEDBs provided by the VSO.

10.2 DBCTL IMS 5.1 Improvements

Table 15 on page 180 lists the major enhancements in IMS 5.1 and indicates those items that are relevant to a DBCTL user. The items with two or three stars are those specifically provided for the DBCTL environment.

Table 15. DBCTL 5.1 Overview

	TM	DBCTL
Processing Cost Enhancements		
N-way data sharing	**	**
Fast Path enhancements • VSO • DEDB enhancements • High speed reorganization • Enhanced HSSP • HSSP ASIC	**	*
Work Load Manager	**	*
DB2 Pseudo-WFI	*	
DB2 Program Call Usage	*	
Enhanced Log Formatting and Select Utility	*	*
VSCR in CSA (around 100KB)	*	*
OSAM DCME support	*	*
Operating Cost Enhancements		
New AO facility	*	***
MTO facilities • Database group for database commands • DBALLOC/NODBALLOC parameter • Command message suppression • Command language modification facility	*	
DBCTL CRC alternative	*	***
Availability and Remote Site Contingency		
Remote site recovery	***	*
Enhanced timestamp recovery	*	*
Enhanced Message Control/Error Exit (DFSCMUX0)	*	
MSC improvements	*	
Dynamic IMS Type 2 SVC update	*	*
Openness and Distributed Processing	*	
Miscellaneous		
Database RACF protection	*	*
Up to 999 dependent regions or CICS threads	*	**
Four-digit device addresses	*	*
Change to /DIS AREA command	*	*
Reduced logging at system checkpoints	*	
Specifying IMS LU name at startup	*	
DBCTL reserved words		*
Dropped Local DL/1 support		*
Dropped LU6.1 adapter support	*	

Chapter 11. IMS 5.1 N-Way Data Sharing

The ability to access common data from many systems, with full read and update integrity, is a key requirement for modern systems. IMS 5.1 extends the previous capability of sharing an IMS database between two MVS systems (called BLDS) and allows sharing by as many as 32 separate systems. This n-way data sharing is likely to be most important in the parallel transaction server (PTS) environment.

An overview of the PTS considerations for IMS 5.1 is provided in this chapter. For a more detailed description and greater technical detail refer to *Using IMS 5.1 with the Parallel Transaction Server,* GG24-4303.

11.1 Reasons to Use N-Way Data Sharing

N-way data sharing can lead to substantial benefits in many different scenarios. Some of these scenarios are subtle and need explanation.

The primary scenarios are:

1. Capacity greater than a single system
   - Need single system image
   - Need data consistency
   - Need future growth
2. Continuous operation
3. Conflicting performance requirements by workload
   - Batch buffering versus online
   - Dataspace or hyperspace usage
4. Phased change control
5. Backup (redundant components)
6. Increased availability
7. Flexible operation
8. Operational error avoidance
9. System managed processing.

The redbook *Using IMS 5.1 with the Parallel Transaction Server,* GG24-4303 gives a detailed examination of the features and implementation of the parallel sysplex environment.

Glossary

A

Advanced Program-to-Program Communication (APPC). An implementation of LU6.2 together with an application programming interface (API). Different platforms (MVS, RISC System/6000, OS/2) have different APPC implementations. A common, standard APPC API is defined, called the Common Programming Interface for Communications (CPI-C).

APPC/MVS. An MVS component that supplies APPC services to programs.

area. An area is a partition of a DEDB. It is a single data set, although there may be several identical copies of it (see multiple area data sets). An area contains all segment types.

B

base section. A set of CIs in a DEDB unit of work. It is the place where IMS tries to store root segments and their direct dependents.

C

control interval (CI). A logical block of data in a VSAM data set. VSAM read and write requests are issued against CIs.

control interval update sequence number (CUSN). A field that IMS maintains in every CI in the direct part of a DEDB. It is used to ensure data integrity after an IMS emergency restart in a block level data sharing environment.

conversation. In APPC, when two programs communicate with each other, they are said to be in "conversation." Every request to allocate a conversation specifies the APPC transaction program name.

commit (or sync) point. The point in time when an application program completes the processing of a message or a batch unit of recovery. For a message-driven application, it is the time when the program asks for the next message. For a batch program, it is the result of a CHKP or SYNC DL/1 call. Any changes the program has made to resources can now be made permanent. Any resources the program held can now be released.

Common Programming Interface for Communications (CPIC or CPI-C). An API for APPC that is standard across many environments.

Communications Manager/2 (CM/2). A product that supplies APPC and other communication services to OS/2 programs.

D

database control (DBCTL). One of the execution modes of IMS. In this mode, IMS manages databases only. Terminal and transaction management is performed by an external subsystem such as CICS.

(DMAC). A DEDB control block that contains vital information about an area.

data entry database (DEDB). An IMS Fast Path database that provides high performance, high availability, and support for very large amounts of data. It also includes an extremely efficient data entry facility, after which it is named.

dependent overflow. A set of CIs in a DEDB unit of work, providing first level overflow from the base section. It is the first place where IMS looks for space to insert a segment that will not fit in the most desirable CI in the base section.

See also independent overflow.

direct dependent segment (DDEP). An ordinary type of segment within a DEDB. It is a segment that is neither a root nor an SDEP.

E

Expedited Message Handling (EMH). The message processing component of IMS Fast Path.

Extended Recovery Facility (XRF). An IMS facility for high availability using a hot-standby alternate system.

Extended Terminal Option (ETO). A function of IMS that allocates terminal and user control blocks dynamically at logon and sign-on times. The alternative is that static terminals are predefined in the IMS system definition.

F

Fast Path (FP). A group of IMS functions that are simpler but faster than the rest of IMS′s functions. Fast Path comprises DEDBs, MSDBs, and EMH.

field search argument (FSA). A parameter you pass to a DL/1 Field (FLD) call. It specifies which actions IMS should perform on that field.

front-end switch (FES). A very efficient message routing facility in IMS that allows transactions for

other systems (not necessarily IMS systems) to be forwarded to the appropriate target system. It includes facilities for keeping the input terminal in response mode, specifying a timeout period, and determining what to do in the event a reply comes back after timeout.

Full Function (FF). A collective term for IMS databases that are not part of Fast Path.

H

hierarchical direct access method (HDAM). A type of IMS database where the hierarchical structure is maintained by pointers and a root segment's position is derived by randomizing the root key.

hierarchic indexed direct access method (HIDAM). A type of IMS database that is similar to HDAM but uses an index to access root segments. HIDAM stores root segments in key sequence.

High Speed Sequential Processing (HSSP). An especially fast process that you can invoke to process a DEDB sequentially in batch.

High Speed Sequential Retrieval (HSSR). A separate program product (product number 5787-LAC) that provides faster sequential processing of IMS databases.

Highly Parallel Transaction System (HPTS). A new IBM architecture for processing transactions in a sysplex using a large number of loosely coupled computers. IMS 5.1 supports and utilizes the HPTS environment.

I

independent overflow (IOVF). A part of a DEDB area. It is the place where IMS puts segments after it has filled the appropriate dependent overflow.

intersystem communication (ISC). A method of communication between IMS systems, between IMS and CICS systems, or between CICS systems. IMS ISC uses the LU 6.1 communication protocol.

L

logical unit (LU). A node in the telecommunications network. A logical unit can be a printer, a terminal, a computer, or a program.

Logical Unit 6.1 (LU6.1). An obsolete protocol that programs use to communicate with one another. IMS and CICS use this protocol for ISC. The LU6.1 protocol has been replaced by LU6.2.

Logical Unit 6.2 (LU6.2). An advanced protocol that programs use to communicate with one another (synonymous with APPC). LU6.2 is the preferred protocol for most advanced applications.

M

main storage database (MSDB). A type of IMS database held in main storage. The DEDB VSO option provides a preferred method of holding data in memory.

mode. A VTAM control block describing the attributes of a network connection.

multiple area data sets (MADS). An option for a DEDB area to be replicated with up to seven copies, each of which is an area data set (ADS). This provides very high reliability and availability for the database data content.

Multiple Systems Coupling (MSC). A method of communicating between IMS systems. It is independent of the actual connection mechanism.

N

normal buffer allocation (NBA). For a dependent region, the number of buffers in the Fast Path common pool that programs should normally find sufficient. In exceptional cases, more buffers can be allocated up to an additional maximum called the overflow buffer allocation (OBA). Only one program at a time, in the whole IMS system, can be using its OBA.

O

output thread. An output thread is a separate task that IMS uses to write updated DEDB CIs asynchronously to DASD.

overflow buffer allocation (OBA). The number of extra Fast Path database buffers a region can use above its NBA. IMS does not permit a program to exceed this limit.

P

partner. In APPC one of the two parties in a conversation.

Parallel Sysplex (PS). A new IBM architecture for running a sysplex of loosely coupled computers. Supported by IMS 5.1.

Parallel Transaction Server (PTS). A new IBM computer architecture using CMOS technology which implements a parallel sysplex. Supported by IMS 5.1.

R

randomizer. A program that uses the key of a segment to decide its position in the database. It is much faster than searching via an index. DEDBs and HDAM databases use randomizers.

The randomizer does not select an arbitrary position for the segment; it chooses one of the predefined root anchor points (RAPs).

relative byte address (RBA). A position within a data set, measured in bytes from the start. The first byte has an RBA of zero, the second byte, an RBA of 1 (because it is 1 byte from the start), and so on.

root addressable part. A part of a DEDB area. It is the collective term for all base and dependent overflow CIs.

root anchor point (RAP). A special kind of pointer in a DEDB or HDAM database. A RAP points to a root segment. The randomizer chooses which RAP is assigned to each root.

S

side information. (1) A name (for example, TP name, LU name) in a table of a set of APPC information that identifies an APPC application program. Another application can use this single name to represent the full APPC address.

sequential dependent part. A part of a DEDB area. It is the place where IMS stores SDEP segments.

sequential dependent (SDEP) segment. A special type of segment within a DEDB, stored in the sequential dependent part, that is inserted very efficiently using a standard ISRT call. It is designed to be retrieved in bulk with the SDEP Scan utility, and deleted, to make space for reuse, by the SDEP Delete utility.

subset pointer (SSP). A special kind of database pointer that an application program may set and use. Only DEDBs support subset pointers.

sync point. Another name for commit point.

synonym chaining. A phenomenon of DEDBs and HDAM databases. These databases use a randomizer to select a root anchor point off which to chain the roots.

Sometimes the randomizer places more than one root at the same RAP. These two roots are then said to be synonyms of each other. IMS maintains a chain of pointers from one synonym to the next. This is called a “synonym chain.” IMS maintains the chain in key sequence. The RAP points to the first root in the chain.

See randomizer and root anchor point.

T

transaction program (TP). Another name for an APPC application program.

transaction program instance (TPI). An executable instance of an application program. Each APPC conversation creates a separate TPI.

transcation program name (TPN). The name of the application program to be executed by an APPC allocate request. Often called a TP.

U

unit of work. (1) The smallest amount of application program processing that is internally consistent. All elements of a unit of work are committed or backed out together. (2) In a DEDB, a unit of work is a predetermined number of contiguous CIs in the root-addressable part of the database.

V

virtual storage constraint relief (VSCR). Certain areas of virtual storage are in short supply. Successive releases of IMS and other products have sought to reduce their use of these areas of virtual storage. This is known as virtual storage constraint relief.

Virtual Storage Option (VSO). An option for DEDBs that allows the data to be kept in memory in an MVS data space. Periodically, IMS asynchronously writes any updated CIs back to the area data sets.

W

Wait for Input (WFI). A type of IMS application program that remains in the dependent region even when there is no immediate message to process.

List of Abbreviations

ACB	Access Control Block
ADS	Area Data Set
AIB	Application Interface Block
API	Application Programming Interface
APPC	Advanced Program to Program Communication
APSB	Allocate PSB
ASIC	Asynchronous Image Copy
BBO	Batch Backout
BLDS	Block Level Data Sharing
BMP	Batch Message Driven Program
CF	Coupling Facility
CI	Control Interval
CICS	Customer Information Control System
CPIC (or CPI-C)	Common Programming Interface for Communications
CUSN	CI Update Sequence Number
CSOM	Client Server Object Manager
DBCTL	Data Base Control
DBD	Data Base Definition
DBRC	Data Base Recovery Control
DCCTL	Data Communications Control
DCE	Distributed Computing Environment
DCME	Dynamic Cache Management Enhanced
DDEP	Direct Dependent (Segment)
DEDB	Data Entry Data Base
DLT	Database Level Tracking
DOVF	Dependent Overflow Section
DPSB	Deallocate PSB
ECSA	Extended Common Service Area
EMH	Expedited Message Handling
ESAF	External Subsystem Attach Facility
ESS	External Subsystem
ETO	Extended Terminal Option
FES	Front End Switch
GSAM	Generalized Sequential Access Method
GSG	Global Service Group
GUI	Graphical User Interface
HDAM	Hierarchic Direct Access Method
HIDAM	Hierarchic Indexed Direct Access Method
HPTS	Highly Parallel Transaction System
HSSP	High Speed Sequential Processing
IBM	International Business Machines Corporation
IFP	IMS Fastpath Program (or Interactive Fastpath Program)
ILS	Isolated Log Sender
IMS	Information Management System
IOVF	Independent Overflow Part
IRLM	IMS Resource Lock Manager
ISC	Intersystem Communications
ITSO	International Technical Support Organization
JES	Job Entry Subsystem
LU	Logical Unit
MADS	Multiple Area Data Sets
MFS	Message Format Service
MID	Message Input Discriptor
MOD	Message Output Discriptor
MPP	Message Processing Program
MPR	Message Processing Region
MQI	Message Queueing Interface
MSC	Multiple Systems Coupling Facility
MSDB	Main Storage Data Base
MTO	Master Terminal Operator
NBA	Normal Buffer Allocation
NMD-BMP	Non-message Driven BMP
OBA	Overflow Buffer Allocation
OLDS	Online Log Data Set
OSAM	Overflow Sequential Access Method
OSF	Open Software Foundation

OTHRD	Output Thread
OTMA	Open Transaction Manager Access
PCB	Program Communications Block
PI	Program Isolation
POSIX	Portable Operating Systems Interface
PSB	Program Specification Block
Pseudo-WFI	Pseudo Wait for Input
RACF	Resource Access Control Facility
RAP	Root Anchor Point Root Adressable Part
RBA	Relative Byte Address
RECON	Recovery Control (data set)
RLT	Recovery Level Tracking
RPC	Remote Procedure Call
RSR	Remote Site Recovery
SDEP	Sequential Dependent (Segment)
SG	Service Group
SLDS	System Log Data Set
SLUP	Secondary Logical Unit - Programmable
SMB	Scheduler Management Block
SMU	Security Maintenance Utility
SPA	Scratch Pad Area
SSP	Subset Pointer
STSN	Set and Test Sequence Numbers
SVC	Supervisor Call
TCO	Time Controlled Option
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TM	Transaction Manager
TMS	Transport Manager Subsystem
TPN	Transaction Program Name
UOR	Unit of Recovery
UOW	Unit of Work
USID	Update Set Identifier
VSAM	Virtual Storage Access Method
VSCR	Virtual Storage Constraint Relief
VSO	Virtual Storage Option
VTAM	Virtual Telecommunications Access Method
VTCB	VTAM Terminal Control Block
WADS	Write Ahead Data Set
WFI	Wait for Input
XCF	Cross System Coupling Facility
XES	Cross System Extended Services
XRF	Extended Restart Facility

Index

Special Characters

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

W

X

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
IMS/ESA Version 5.1 Guide
February 1995

Publication No. GG24-4302-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:
   Do you provide billable services for 20% or more of your time? Yes____ No____
   Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name	Address
Company or Organization	
Phone No.	

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 471, Building 070B
5600 COTTLE ROAD
SAN JOSE CA
USA 95193-0001

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.