IMS Version 13 Technical Overview

Explore the features and functions of IMS 13

Understand advantages and applicability of IMS 13

Plan for installation of or migration to IMS 13

Paolo Bruni
Juan Jesús Iniesta Martínez
Jouko Jäntti
Rick Long
Geoff Nicholls

Redbooks

ibm.com/redbooks

International Technical Support Organization

IMS Version 13 Technical Overview

September 2014

SG24-8224-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xxiii.

First Edition (September 2014)

This edition applies to IMS Version 13 (program number 5635-A04).

Contents

Figures

Tables

Examples

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

Preface

IBM® Information Management System (IMS™) provides leadership in performance, reliability, and security to help you implement the most strategic and critical enterprise applications. IMS also keeps pace with the IT industry. IMS, Enterprise Suite 2.1, and IMS Tools continue to evolve to provide value and meet the needs of enterprise customers.

With IMS 13, integration and open access improvements provide flexibility and support business growth requirements. Manageability enhancements help optimize system staff productivity by improving ease of use and autonomic computing facilities and by providing increased availability. Scalability improvements were made to the well-known performance, efficiency, availability, and resilience of IMS by using 64-bit storage.

IBM IMS Enterprise Suite for z/OS® V2.1 components enhance the use of IMS applications and data. In this release, components (orderable or downloaded from the web) deliver innovative new capabilities for your IMS environment. They enhance connectivity, expand application development, extend standards and tools for a service-oriented architecture (SOA), ease installation, and provide simplified interfaces.

This IBM Redbooks® publication explores the new features of IMS 13 and Enterprise Suite 2.1 and provides an overview of the IMS tools. In addition, this book highlights the major new functions and facilitates database administrators in their planning for installation and migration.

Authors

This book was produced by a team of specialists from around the world working at the IBM Silicon Valley Laboratory, San Jose, California.

Paolo Bruni is an International Technical Support Organization (ITSO) Project Leader that is based at the Silicon Valley Lab in San Jose, CA. Since 1998, Paolo authored IBM Redbooks publications about IMS, DB2® for z/OS, and related tools, and conducted workshops worldwide. During his many years with IBM in development and in the field, Paolo’s work was related mostly to database systems.

Juan Jesús Iniesta Martínez is a Certified Consulting I/T Specialist with IBM Global Services in Spain. He has 27 years of experience in the IMS field. He holds an IBM IMS Certificate of Achievement of Mastery in Fundamentals, Systems Programming, and Database Administrator. Before joining IBM in 1995, Juan worked for eight years as an IMS Systems Programmer in a large bank in Spain. Since Juan joined IBM, he was assigned to several large banking clients, providing IMS support and guidance. Juan manages the Spanish IMS GUIDE (GSE) users working group since 2003. His main areas of expertise include IMS installation, Parallel Sysplex® architecture, data sharing, IMS shared queues, and IMSplex installations. Juan co-authored several IBM Redbooks of IMS on Parallel Sysplex and technical overviews of new IMS versions.

Jouko Jäntti is a Certified Senior IT Specialist in Finland. He has 28 years of experience in IMS field, and worked with all the IMS versions since IMS 1.3 up to IMS Version 13. Before joining IBM, he worked as a systems programmer in a large Finnish bank. He worked at IBM for 17 years, part of that time as a project leader for ITSO IMS Redbooks during the years 2001 - 2007. He is a lead author for 15 IMS related Redbooks, and was a co-author for two

IMS Redbooks. After the time in ITS, Jouko was a member of IMS Worldwide Technical Specialist Team (formerly IMS Worldwide Advocate Team).

Rick Long is an IMS level 2 database support specialist in the Silicon Valley Lab IMS level 2 database support team. He was an IMS systems specialist at the ITSO in San Jose. Before working in the ITSO, he worked in the DBDC programming department of IBM Australia as an IMS systems programmer and IMS Database Administrator. He also worked for several IBM customers as an IMS application programmer. He writes extensively and teaches IBM classes worldwide on all areas of IMS. Rick is a co-author of *An Introduction to IMS, Your complete Guide to IBM's Information Management System* (ISBN-10: 0-13-185671-5 and ISBN-13: 978-0-13-185671-4).

Geoff Nicholls is a Consulting IT Specialist, working in the IMS Solution Test team for the IBM Silicon Valley Laboratory, and is based in Melbourne, Australia. Before his current role, Geoff was a member of the Worldwide IMS Advocate team for 12 years, providing consulting, education, and services to clients in many industries around the world. Geoff is the co-author of 13 IMS IBM Redbooks publications. Before joining IBM, Geoff worked as an Applications Programmer and Database Administrator for several insurance companies and another mainframe vendor. Geoff has a Bachelor of Science, majoring in Computer Science, from the University of Melbourne.

Special thanks to all of the people who contributed to the preparation of the material that was used for IMS 13 introduction to marketplace and let us use it in this effort.

Thanks to the following people for their contributions to this project:

Carlos Alvarado
Jim Bahls
Tom Bridges
John Butterweck
Himakar Chennapragada
David Compton
Chad Deluca
Haley Fung
Bryan Hickerson
Kevin Hite
Deepak Kohli
Terry Krein
Rose Levin
Evgeni Liakhovitch
Janna Mansker
Bruce Naylor
Khiet Nguyen
Betty Patterson
Richard Schneider
Patrick Schroeck
Nai-Wen (Rita) Shih
Anu Vakkalagadda
Scott Ward
Jack Yuan
IBM Silicon Valley Laboratory

Kenneth Blackman
Dennis Eichelberger
Glenn Galler
Nancy Stein

Suzie Wendler
IBM IMS Advanced Technical Skills Americas

Gary Wicks
IBM Markham

Janet Leblanc
IBM Ontario

Tatsuyuki Shiomi
IBM Tokyo

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at this website:

http://www.ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

IMS 13 at a glance

Built on the trusted IBM System z® platform, Information Management System (IMS) 13 helps enterprises successfully make data available anywhere throughout the organization and succeed in a changing landscape.

IMS set the standard for performance, reliability, and scalability. With the release of IMS 13, IBM builds on the success of IMS open-access technologies, which makes IMS data available to any platform or application across the enterprise. IMS helps you grow your business, whether your goals are toward big data, analytics, or mobile, and increases the company's business agility while delivering more performance and more savings.

IMS 13 features the following enhancements:

- Empower IT organizations to integrate critical IBM IMS data across the enterprise
- Enable trusted data and transactions to be reused for innovative web services and mobile implementations
- Reduce application development time and costs through simplified interfaces and tools
- Dramatically improve throughput, performance, and productivity

This chapter includes the following topics:

- What is new in IMS 13
- Leading online transaction and database management software
- Delivering next-generation cost per performance
- Open, flexible IMS 13 is the right fit for the future
- Turning your trusted applications into tomorrow's web services

1.1 What is new in IMS 13

With IMS 13, IBM enhanced the proven, rock-solid IMS technology with new functionality that is driven by today's evolving business requirements (see Table 1-1), as shown in the following examples:

- Online Alter function for IMS High Availability Large Databases (HALDB) and Fast Path data entry databases (DEDB) simplifies database changes and enhances availability.
- Database versioning offers greater flexibility for implementing database changes and speeding new program deployments without the need to roll out changes to all existing programs.
- By using IBM IMS Enterprise Suite Data Provider for Microsoft .NET, you can use standard SQL queries to access IMS data from .NET applications.
- Native SQL support for COBOL enables SQL in COBOL programs to access IMS databases and provides SQL processing natively in IMS.
- IBM IMS Connect, the TCP/IP gateway to IMS transactions, operations, and data offers improved IMS flexibility, availability, resilience, and security.
- TCP/IP support for Intersystem Communication (ISC) links between IMS and IBM CICSR can help reduce maintenance costs and complexities.
- Significant enhancements led to the IMS record-setting performance metric of over 100,000 transactions per second.

Together, IMS 13 and enhanced IMS Tools enable businesses to reduce costs, boost productivity, and easily grow to meet new demands.

Table 1-1 IMS 13 enhancements by area

Database Manager	**Transaction Manager and connectivity**	**Systems Manager**	**Enterprise Suite**
HALDB Alter enhancement	DL/I ICAL call and Java API enhancements for synchronous program switch	Reduced total cost of ownership enhancements	.NET Data Provider for IMS DB
DEDB Alter enhancement	Asynchronous callout enhancements for WebSphere® MQ	Application thread increase and development enhancements	IMS SOAP Gateway enhancements
Database versioning	Several OTMA enhancements	IMS security enhancements	IMS User Interface enhancements
Native SQL for COBOL	IMS Connect enhancements	Stand-alone security user exit	N/A
DBRC enhancements	ISC over TCP/IP	IMS repository enhancements	N/A
/START DB command enhancement	IMS shared queues local first enhancement	ESAF support in Java dependent regions enhancement	N/A
IMS Catalog Populate utility (DFS3PU00) serviceability enhancements	DRDA® DDM command support for native SQL enhancement	/DIAGNOSE command enhancements	N/A

Database Manager	Transaction Manager and connectivity	Systems Manager	Enterprise Suite
SQL support enhancements	N/A	Extended address volume (EAV) enhancement for GSAM	N/A
N/A	N/A	Command enhancements	N/A
N/A	N/A	IMS CQS enhancement	N/A
N/A	N/A	User exit enhancements	N/A

1.2 Leading online transaction and database management software

Designed to handle mission-critical applications, IMS is a key element of IBM z/OS middleware and provides the backbone for much of the world's enterprise computing. IMS offers a *gold standard* for transaction processing performance and scalability. Customers routinely handled peaks that exceed 100 million transactions in a day. It also provides high availability, with customers experiencing as many as 3,000 days with no unplanned outages.

IMS 13 is made up of the IMS 13 Database Manager for data and batch processing, and the IMS 13 Transaction Manager for transaction and batch message processing. Both are simple to install and use. With the latest improvements, both deliver throughput and performance enhancements that enable your organization to continue the use of your IMS and z/OS investments well into the future.

1.2.1 Multiple versions of an IMS database

Database versioning support is for Full Function, HALDB, and DEDB database customers who need support for multiple views of the physical data to various application needs, including the following examples:

- Implementing application changes over time.
- Ability to use application programs (for which there is no source code) after database structure changes.

IMS database versioning supports databases that include logical relationships and databases that include secondary indexes.

The database versioning function can be used with the database alter functions to track different versions of the structure of a database. It supports the following structural changes to all supported database types:

- Increasing the length of a segment.
- Adding a field at the end of a segment

These changes are normally implemented by recoding the database description (DBD) source and running the DBD, PSB, and ACBGEN utilities. The customer then unloads or reloads the database or uses the IMS 13 HALDB Alter function, followed by performing an Online Change (OLC).

Changes made to any existing fields (including changing the starting position or length of the field) are not allowed. Deleting fields is not supported by versioning.

Database versioning provides the ability to assign user-defined version identifiers to different versions of the structure of a database. By using these identifiers, you can make structural changes to a database while providing multiple views of physical data to various applications.

New applications that reference a newer structure of a database can be brought online without affecting applications that use previous database structures. Unchanged applications, which do not have to be sensitive to the new physical structure of the database, can continue to access the database.

Versioning support enables users to assign user-defined version IDs to different versions of the structure of a database. The user-defined version IDs are stored in the record for the database in the IMS Catalog. Upon accessing the database, application programs specify the version of the database that they need. If they do not specify a version, by default they receive the version of the database structure at the current level.

Versioning support for databases uses metadata to provide multiple views of data to applications. Metadata is created by ACBGEN when the ACB member is created. Metadata is in the IMS DB Catalog. The use of the IMS Catalog is required. Metadata in the catalog identifies the view of the existing data on the data set. There can be multiple versions of the metadata, one for each application view. There can be multiple applications that are accessing this data that requires different views of this data.

Data is read from the database. This data is transformed into the view that is wanted by the application by using two versions of metadata: one describes the segment on the database and the other the application view. The transformed segment is put into the users application I/O area.

Supported DB types include HDAM, HIDAM, PHDAM, PHIDAM, and DEDB

The following DFSDFxxx PROCLIB member keywords are used:

- DBVERSION=

  Indicates whether to use versioning. Default is not to use versioning.

- DBLEVEL=

  Determines behavior for versioning. Default is to always use current physical structure, so application always gets current DBD. Option for base DBD for those that never change their applications and do not want them affected by DB changes.

The following database and program generation statements are used:

- DBD DBVER=

  Incremental version number that is specified by the customer when the DB is changed.

- PCB DBVER=

  Version number the PCB wants to use for the specific DB.

- PSBGEN DBLEVEL=

  Default version for all PCBs in the PSB. Can specify CURR for current level or BASE for the base level.

- DL/I Call: INIT VERSION

  Can be used once before any DB call to set the required DB version for the application. This overrides the PCB DBVER definition.

Database versioning requires IMS catalog enablement because the DBD version definitions are stored in the IMS catalog.

The IMS catalog metadata describes the current and previous structures of a database. The metadata includes the version numbers that identify each structure of a database. When an application program makes a call to a versioned database, IMS internally references the catalog to determine which structure corresponds to the provided version number and whether the format of the requested data must be modified before it is returned to the application program.

To enable the database versioning support, you must specify the new parameter, DBVERSION=Y, in the databases section in the DFSDFxxx member of the IMS PROCLIB data set to indicate that database versioning is to be used.

An application program that must access a particular version of a database definition can specify the DBVER= on the PCB statement of the PSB source, or issue the INIT VERSION call to specify the database version for each database view that is used by the application.

For more information about versioning, see Chapter 6, “Database versioning” on page 181.

1.2.2 HALDB Alter

You can change the physical structure of an IMS High Availability Large Database (HALDB) without having to unload and reload the database. HALDB Alter extends HALDB Online Reorganization to allow changes to add a field to the end of a segment and increase the length of a segment.

Structure changes are performed while the database is online. Online Reorganization (OLR) is used to transform a database from one version to another. While OLR is running, there are two versions of the physical data. The old version in the input data set and the new version in the output data set.

After OLR is complete, there is only one physical version of the data on the data set; the highest version of the data.

Basic OLR command specifies one or more partition names. ALTER specifies a single master DB name. You cannot specify the partition name as the entire DB must be altered before the new version can be reused. QUERY OLREORG shows the status.

Before a HALDB is altered, you can specify the new partition block sizes (OSAM) or CI sizes (VSAM) by using the new ALTERSZE keyword. These new sizes cannot be changed if the HALDB is being altered. DBRC command CHANGE.PART DBD(name) PART(name) ALTERSZE(nnnnn).

> **Note:** The RECON partition record now contains CI sizes of VSAM data sets. The BLOCKSZE keyword is now allowed with VSAM.

The `CHANGE.DB` and `NOTIFY.REORG` commands add the capability to set the alter status in their associated RECON records.

LIST output for HALDB Master is changed to show the Alter count and Alter complete count

The user must perform an Online Change (OLC) before the structure changes can be used. The OLC does require the database to be taken offline by using a **/DBR** or **UPDATE DB** command while the change occurs. This is needed to allow users to control when the applications see the changes and to coordinate with installation requirements for change management.

For more information about HALDB Alter, see 4.1, “HALDB alter” on page 102.

1.3 Delivering next-generation cost per performance

Operational efficiency is critical. Organizations are doing more with fewer personnel, which drives down internal and external costs and provides maximum returns. IMS 13 delivers significant double-digit (and in some cases triple-digit) gains in performance, throughput, and productivity. You can expect to see savings from various efficiencies, including reduced cost of ownership, improvements in overall performance and throughput, and enhancements that are designed to save time and boost productivity.

In this section, we provide a summary of recent performance measurements and improvements. For more information, see Chapter 9, “IMS performance enhancements” on page 231.

1.3.1 Lower total cost of ownership

IBM System z can lead to CPU savings and improved operational efficiency. IMS 13 is built on System z, which continues to offer the highest levels of virtualization in the industry for a more efficient data center that requires less power, cooling, and space. Synergy with other System z platform components reduces CPU use by using the latest processor improvements, increased memory, solid-state disk technology, and z/OS software enhancements. Cost-effective zIIP solutions for some IMS workloads are shown to lower CPU usage by over 22%.

System z Integrated Information Processor (zIIP) specialty engine support is extended to optionally include some IBM authorized processing in the IMS Connect address space and in the IMS Open Database Manager (ODBM) address space. This includes selected processing in IMS Connect that is related to SOAP Gateway, Distributed Relational Database Architecture™ (DRDA), Multiple System Coupling (MSC), and InterSystem Communication (ISC). Also included is selected processing in ODBM for DRDA and CSLDMI API requests.

The percentage comparison values reported are percent reductions in GP CPU microseconds per transaction between the baseline (zIIP offload disabled for all IMS address spaces) and zIIP offload enabled (for the IMS Connect and ODBM address spaces). Any user exits called by this processing is not called while it is running under an enclave SRB. User exits are always given control in TCB mode. Also, because of technical restrictions, certain processing cannot run under enclave SRBs. Such processing includes calling z/OS Resource Recovery Services (RRS), IMS DL/I call processing, and z/OS supervisor calls (SVCs). IMS switches from SRB mode into TCB mode to perform such processing, and thus, such processing does not run on a zIIP.

When enclave SRB execution is enabled, IMS 13 directs z/OS to authorize certain work to be processed on an available zIIP. Portions of the following processing can run under an enclave SRB in IMS 13:

- The processing of IMS Connect and ODBM address space Distributed Relational Database Architecture (DRDA) threads for DRDA requests that are arriving via TCP/IP (DRDA workload)
- The processing of IMS Connect address space SOAP message threads for SOAP messages that are arriving via TCP/IP (SOAP workload)
- The processing of IMS Connect address space Multiple System Communication (MSC) threads for MSC messages that are arriving via TCP/IP (MSC workload)
- The processing of IMS Connect address space Intersystem Coupling (ISC) threads for ISC messages that are arriving via TCP/IP (ISC workload)
- The processing of ODBM address space threads for requests that are arriving through the CSLDMI[1] API (CSLDMI workload)

An IMS benchmark was run with zIIP utilization enabled [CONDSRB(ALWAYS)] and disabled [CONDSRB(NEVER)] in the BPE configuration PROCLIB members for the address spaces that use zIIPs (IMS Connect and in the case of the DRDA workload ODBM).

The average number of CPU seconds that are used per transaction (GP and zIIP) was used to normalize the comparisons. The following percentages were measured to show the reduction in general CPU time by moving work to zIIP engines:

- MSC workload 4.58%
- ISC workload 4.85%
- SOAP workload 8.47%
- DRDA workload with RRS=N 22.77%
- DRDA workload with RRS=Y 15.21%

1.3.2 Increased throughput and performance

IBM is dedicated to continuous improvement of IMS, which ensures the highest-possible levels of performance, scalability, and availability. With IMS 13, IBM engineers collaborated across products to help IMS achieve a record-setting Fast Path performance benchmark of 117,292 transactions per second, which is an 800% increase over IMS 12.

That work also led to reduction or elimination of other system bottlenecks, which results in higher performance and efficiency across IMS configurations. For example, logging enhancements reduced contention by 88 percent.

These capabilities help businesses become more resilient, reduce business risk, and comply with regulations and business processes.

1.3.3 Enhanced productivity

Integrated IMS Tools and the IMS Enterprise Suite enable IT architects and developers to save time and increase their productivity. IMS Enterprise Suite provides easy-to-use IMS interfaces for application development, installation, operation, and reporting, which greatly reduces the need for special, in-depth IMS skills. It includes the IMS Data Provider for Microsoft .NET, which handles the connections and communications with IMS through IMS Connect by following the IBM Distributed Relational Database Architecture (DRDA) protocol.

[1] Common Service Layer Database Manager Interface

Microsoft .NET application developers can use their preferred development environment, such as Microsoft Visual Studio, to call the provided application programming interfaces (APIs) to easily access IMS databases. The IMS Tools base delivers the IMS Explorer for Administration, which provides a web-based GUI for standard IMS systems management tasks.

1.4 Open, flexible IMS 13 is the right fit for the future

IBM understands that many large enterprises have complex IT infrastructures. IMS 13 is designed to reduce this complexity with extensive integration capabilities that continue to provide strategic value. For example, IBM Cognos® business analytics and reporting can be run directly against IMS data, so there is no need to offload the data to a relational database management system.

IMS 13 integrates with various IBM business software, including Cognos and the IBM InfoSphere® portfolio. These applications are at the heart of many of today's business information systems. IBM InfoSphere Guardium® provides a simple, robust solution for assuring the privacy and integrity of trusted information in your IMS database. Cognos Business Intelligence sets the standard for reporting and analysis. These solutions continue to evolve together with IMS, which offers a path to innovation that can help your company seize the most promising growth opportunities ahead.

With so much of today's commerce driven by the web, the capability of IMS 13 to simplify and enhance service-oriented architecture (SOA) and other web solutions can be critical to your organization. IMS 13 supports Web 2.0 for lightweight integration and rapid web application assembly. It enables SOA environments with reuse of trusted applications and data, industry-standard tools, and interfaces. It extends open access with technological advancements that enrich functionality, optimize performance, and support strategic initiatives, such as mobile, cloud, and big data.

For more information about IMS 13 connectivity enhancements, see Chapter 8, "IMS Connect enhancements" on page 215. For more information about accessing IMS data from various clients, see *IMS Integration and Connectivity Across the Enterprise,* SG24-8174, and *IBM Cognos Business Intelligence 10.2.0 Reporting on IMS,* REDP-5091.

1.5 Turning your trusted applications into tomorrow's web services

IMS is IBM's premier transaction and pre-relational database management system, which is almost unsurpassed in database and transaction processing availability and speed. For decades, IMS clients trusted IMS with their most important business asset; that is, their operational data.

IMS 13 extends those trusted capabilities and assets into the future, with enhanced capabilities that enable you to use your valued core infrastructure to meet new challenges. IMS connectivity enables organizations to integrate IMS across the enterprise. IMS applications and transactions become web services, which enable your business to attract new customers and keep ahead of the competition. At the same time, you can avoid the cost and risk of migrating and rewriting applications by using the performance, reliability, and scalability you trusted for years.

1.5.1 IMS user interfaces

IMS is making several major changes to ease IMS application development and system administration. Typically, development and administration of IMS was done by using ISPF green screens, but IMS now has two state-of-the-art graphical user interface solutions to help your new DBAs, system programmers, and developers get up to speed quickly with IMS, as shown in Figure 1-1.

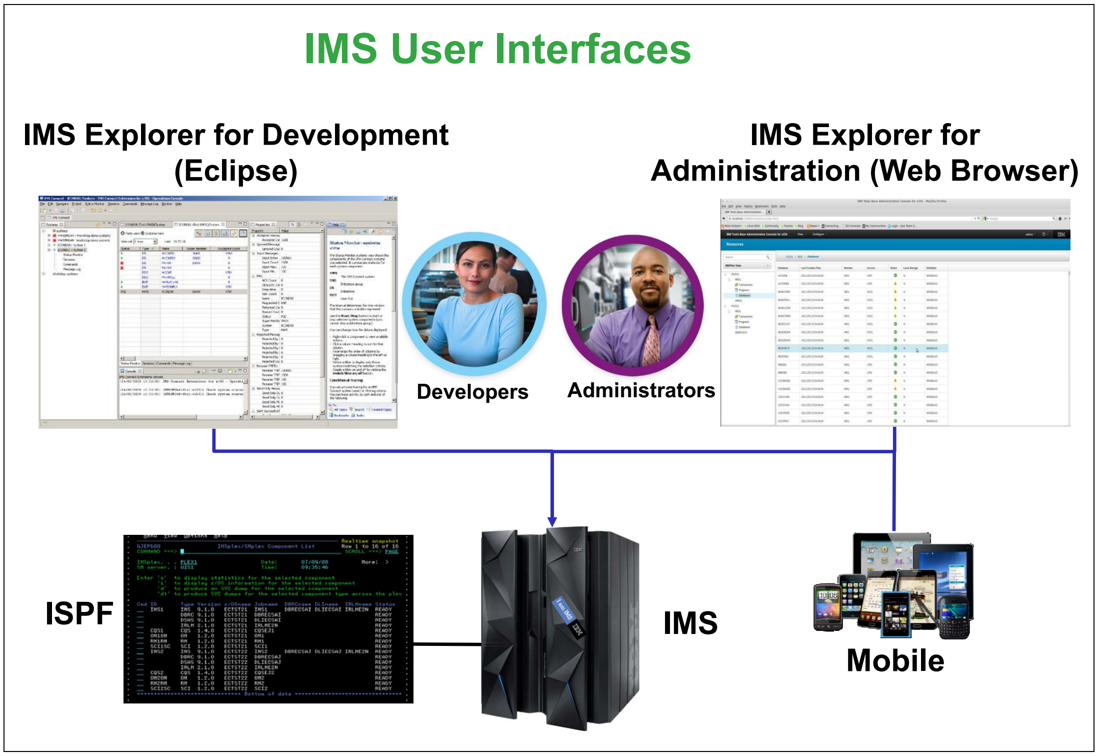


Figure 1-1 IMS user interfaces

The following solutions are available:

- The IMS Explorer for Development

  A graphical tool that is based on the Eclipse platform, IMS Explorer for Development offers capabilities to import, visualize, and edit IMS DBDs and PSBs. By using this tool, you can display the segment hierarchy for any IMS database, including logical relationships and secondary indexes.

  By using the universal JDBC driver, the IMS Explorer for Development offers the graphical tooling with which a user can point-and-click and automatically generate SQL statements. Not only can the user query and manipulate data, but they also can use the statements on an application that uses SQL to access IMS DB, submit JCL, and inspect output in JES. This mode of SQL building via a tool and copying into code is a standard usage pattern for application developers.

  The IMS Explorer for Development version in IMS Enterprise Suite 3.1 also adds the IMS Catalog Navigation view and several built-in queries to assist with resource and relationship discovery.

  By using the new feature IMS Transaction Unit Test support, you can create a transaction test project in which you define a transaction and import the IMS COBOL or PLI application data structures to specify the layout of the input message as expected by the IMS application and the output message that is created by the IMS application.

- The IMS Explorer for Administration

  The IMS Explorer for Administration is a replacement for IMS Control Center. It is introduced as a web-based GUI console for IMS system management. The IMS Explorer for Administration can query, start, and stop IMS resources from the browser interface.

  It can discover IMS systems and their associated resources, show the health of the resources, and show relationships between them.

  IBM IMS Explorer for Administration is available through APAR PM94292 as an extension of the Administration Console component of IBM Tools Base for z/OS, V1.4.

  IMS Explorer for Administration can show IMS systems and resources in an easy-to-read display. Tasks can be performed by using menu pull-downs and right-clicking resources. Extensive help information is provided through a combination of hovering over text and selecting certain text fields.

1.5.2 Native SQL support for COBOL

SQL access to IMS database was available to Java applications for many years. Now, COBOL applications can also use SQL to access IMS database expanding IMS database usage to a wider group of application developers. The use of COBOL SQL to IMS database requires IBM Enterprise COBOL for z/OS 5.1 and APAR PM92523.

This function simplifies application programming to allow SQL programmers to access IMS database in a similar method to what is used for relational databases, which expands IMS database usage to a wider group of application and database developers by using SQL skills without requiring in-depth IMS database knowledge.

For more information and a sample application, see 7.2, “Using SQL in COBOL to access your IMS databases” on page 204.

1.5.3 .NET Access to IMS DB

IMS 13 makes available IMS Data Provider for Microsoft .NET. This function is designed to provide standard SQL access to IMS data from Microsoft .NET applications.

Microsoft .NET applications can use an API to transparently and directly read and manipulate IMS data similar to an Open DB solution. Currently, Microsoft .NET applications have access to IMS TM through IMS SOAP Gateway.

The IMS Data provider is built on IMS 13 capabilities and requires the IMS Catalog, IMS Connect, the Common Service Layer (CSL) components Structured Call Interface (SCI), and Open Database Manager (ODBM).

For more information about this function, see 11.6, “IMS Data Provider for Microsoft .NET” on page 360.

1.5.4 Synchronous program switch

IMS 13 introduces a feature that is called synchronous program switch, which can help you modernize your IMS application infrastructure and reduce unnecessary network traffic.

This feature enables synchronous communication between IMS application programs that are running in separate IMS Database/Transaction Manager (DB/TM) or IMS TM-dependent regions.

The new method relies on simpler programming logic than the asynchronous programming switch method that IMS had for many years. Clients can use synchronous program switch to simplify their application programs and to easily use the services of existing IMS application programs.

For more information about this function, see 7.1, "IMS synchronous callout support" on page 198. Other samples are available in REXX and PLI.

2

System enhancements

This chapter describes the system enhancements for the overall IMS 13 system, including both the IMS Database Manager and the IMS Transaction Manager.

This chapter includes the following topics:

- IMS repository enhancement
- IMS user exit enhancements
- IMS SECURITY macro removal
- RACF password phrase support
- DIAGNOSE command enhancements
- Concurrent application threads
- Reduced total cost of ownership
- IMS zIIP utilization enhancements
- ESAF support in Java dependent regions
- IMS 13 service process changes

2.1 IMS repository enhancement

The resource and descriptor definitions that are supported by dynamic resource definition (DRD) can be stored in the IMS resource definition (IMSRSC) repository. The IMS repository function in IMS 13 supports the IMS change list.

IMS uses the change list to preserve and restore the changes that are made to the IMSRSC repository while the IMS is down.

2.1.1 Highlights

This enhancement includes the following highlights:

- Improved documentation and samples.
- Runtime definition changes can now be applied to an IMS when it warm starts or emergency restarts.

  Changes that are made to active systems by using the **IMPORT SCOPE(ALL)** command (which included resource and descriptor names of an inactive IMS) are later applied to that IMS when it is restarted.

This enhancement simplifies IMSplex management and makes it easier to maintain the synchronization of IMS systems' resources in a DRD with repository environment.

2.1.2 Improved documentation and samples

The process of the use of the DFSURST0, DFSURCM0, and DFSURCL0 utilities, which create RDDS data sets from SYSGEN, MODBLKS, and log records, with the CSLURP10 utility that is used to load RDDS data sets into an IMS Repository Server was unclear and needed sample JCL and procedures.

This enhancement provides sample jobs and procedures for loading the Repository Server from IMS log records, MODBLKS, and SYSGEN. Table 2-1 lists these sample JCL and procedures.

For more information about the procedures that are used in IMS environments, see *IMS 13 System Definition*, SC19-3660.

Table 2-1 Sample JCL and procedures to load the Repository Server

Name	Function
CSLURLFL	Procedure to load Repository Server from log records
CSLURLFM	Procedure to load Repository Server from MODBLKS
CSLURLFS	Procedure to load Repository Server from SYSGEN
CSLURJL0	Sample JCL for starting the CSLURLFL procedure
CSLURJM0	Sample JCL for starting the CSLURLFM procedure
CSLURJS0	Sample JCL for starting the CSLURLFS procedure
CSLURST2	Create the STAGE2 to RDDS to catalog load job
DFSURST0	Updated to support NOMODBLKSHLQ parameter

2.1.3 IMPORT command with SCOPE( )

When you issue an **IMPORT DEFN SOURCE(REPO)** command, you indicate whether it should apply to all IMS systems, including active and inactive systems or only the active IMS systems in the IMSplex, by using the SCOPE() parameter. This parameter indicates to which IMS systems the IMPORT applies.

Figure 2-1 shows the format of the **IMPORT DEFN** command.

```
IMPORT DEFN SOURCE()
TYPE() NAME() OPTION() SCOPE()
```

Figure 2-1 Format of the IMPORT DEFN command

OM API ROUTE capability

You might be familiar with the ROUTE capability of the OM API, which is used to route commands to specific IMS systems.

ROUTE=ALL is recommended when SCOPE(ALL) is included.

If a ROUTE list is specified (other than ROUTE=ALL), the command is processed only by the IMS systems in the list that receive the command.

Other IMS systems that have the resources that are defined but are not included in the ROUTE list do not receive the command and therefore, are not synchronized with the repository.

SCOPE(ALL) option

The SCOPE(ALL) option applies the **IMPORT** command to all active and inactive IMS systems in the IMSplex for which the resource is defined. It is recommended (default) to maintain synchronized definitions across the IMSplex that match the repository definitions.

Inactive systems

If an IMS system is inactive, an IMPORT with SCOPE(ALL) results in an IMS change list that is created for it in the repository, which contains a list of the resources the IMS imports if it was active.

The IMS change list contains a subset of the IMS resource list, which reflects the resources that were not imported because of the IMS being inactive.

When the inactive IMS warm starts or emergency restarts, it applies the IMS change list and imports the resources that are specified in it, which maintains synchronization with other active IMS in the IMSplex that completed the import when the command was issued.

Scenario to apply

This scenario especially applies to a cloned environment where an EXPORT with SET(IMSID(*)) was issued, which updates all IMS resource lists that are associated with IMS that are active and inactive.

A subsequent IMPORT with SCOPE(ALL) then reads in all of these definitions that were updated by the previous EXPORT and maintains definitional synchronization across the multiple cloned IMS systems.

2.1.4 IMPORT command with SCOPE(ALL)

This section describes what happens when an IMS is inactive and an IMPORT with SCOPE(ALL) is issued.

Use IMPORT DEFN SOURCE(REPO) … SCOPE(ALL) to read definitions into all IMS systems in the IMSplex (active and inactive).

The inactive IMS system recalls the IMS change list that is created for it when this command is issued.

Active IMS systems

The active IMS systems imports the stored definitions into the control region from the repository. These definitions become runtime resource definitions in the control region.

Inactive IMS systems

The inactive IMS system has an *IMS change list* that is created for it within the repository. An IMS change list contains a subset of its IMS resource list names that it imports if it was active at the time that the command was issued.

Internally import the updated stored resource definitions at warm start or emergency restart. These definitions become runtime resource definitions in the control region.

The inactive IMS system has its IMS change list deleted at the end of restart after the log records processed, or if it cold starts because autoimport reads its updated stored definitions.

IMS warm starts or emergency restarts

When this inactive IMS warm starts or emergency restarts, it applies the IMS change list and imports the resources that are specified in it, which maintains synchronization with the other active IMS systems in the IMSplex that completed the import when the command was issued.

IMS cold starts

If the IMS cold starts, the IMS change list is deleted and the IMS reads its entire resource list via automatic import.

2.1.5 IMPORT command usage with SCOPE(ALL)

Figure 2-2 on page 17 shows the use of the `IMPORT` command with SCOPE(ALL); however, IMS2 was inactive when the `IMPORT` command was entered and a change list for IMS2 was created.

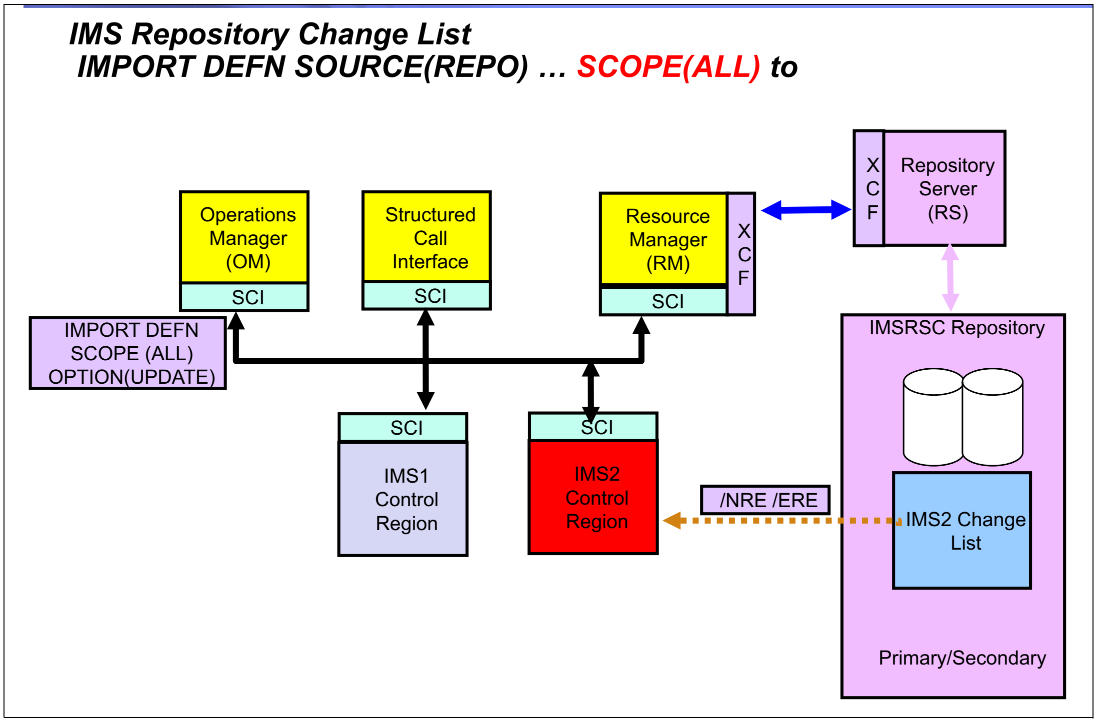


Figure 2-2 Use of the IMPORT command with SCOPE(ALL) without IMS2 active

When IMS2 is started via normal or emergency restart, changes are automatically imported so that the IMS is synchronized with all other IMS systems in the IMSplex.

If a change list is created in the repository, a response line is returned on the **IMPORT** command for each resource name in the change list with the IMSID of the IMS for which the change list is created, along with the import type (IMPTYPE) of the change list.

Note: A change list is created only if the command master IMS is IMS Version 13 or later and the RM that processes change list requests is at V13 level or higher.

IMPORT command without the OPTION(UPDATE) keyword

If the **IMPORT DEFN SCOPE(ALL)** command is issued without the OPTION(UPDATE) keyword and routed to an IMS system that has one or more resources or descriptors that are defined, the **IMPORT** command results in a nonzero return code.

The import might be successful at other active IMS systems, and the command might be successful in creating the change list in the IMSRSC repository for inactive IMS systems.

To avoid a nonzero return code and reason code from the **IMPORT DEFN SCOPE(ALL)** command, complete one of the following tasks:

- Specify the **IMPORT DEFN SCOPE(ALL)** command with the OPTION(UPDATE) keyword and route it to all IMS systems (ROUTE(*)).
- Route the **IMPORT DEFN SCOPE(ALL)** command to active IMS systems in which the resource or descriptors do not exist.

2.1.6 Messages during IMS change list processing

A successful IMS change list processing sequence results in the messages issued that are shown in Figure 2-3.

```
DFS4408I REPOSITORY CHANGE LIST PROCESSING INITIATED
DFS4414I REPOSITORY CHANGE LIST PROCESSING INITIATED FOR DESCRIPTOR
                NAME=descname TYPE=desctype
DFS4410I REPOSITORY CHANGE LIST PROCESSING SUCCEEDED FOR RSCTYPE
               rsctype COUNT count
                           .
                           .
                           .
DFS4410I REPOSITORY CHANGE LIST PROCESSING SUCCEEDED FOR RSCTYPE
               rsctype COUNT count
DFS4412I REPOSITORY CHANGE LIST PROCESSING COMPLETED
```

Figure 2-3 Messages that are issued in a successful IMS change list processing

IMS issues message DFS4408I to indicate that change list processing started.

IMS issues message DFS4414I for every descriptor to be imported from the change list, including the descriptor name and type.

If the change list is successfully processed, IMS issues the DFS4410I message for each resource and descriptor type it successfully processes, including a count of the total number of resources or descriptors that were processed for that particular type.

Because there are four resource types and four descriptor types, there can be up to eight DFS4410I messages issued here.

If the import from the change list fails (DFS4411E message appears), the descriptor is out of sync with the stored resource definitions in the IMSRSC repository.

If there are no errors, the DFS4411E message is not issued. Instead, message DFS4412I that indicates that change list processing is completed is issued.

Note: None of these DFS messages are issued if the IMS is restarting without the IMSRSC repository enabled.

2.1.7 Error scenarios during IMS change list processing

In this section, we describe the following possible error messages when the change list is imported into IMS:

- DFS4401E for an import error

  The DFS4401E message changed to indicate the error return or reason code for the error in internal import from the IMS change list.

  It indicates that the attempt to delete the change list failed. If the change list exists, it is processed the next time IMS warm starts or emergency restarts, or deleted the next time IMS cold starts.

- DFS4411E for an IMS change list processing error

  The DFS4411E message is new in IMS 13 and indicates the error return or reason code for the error in internal import from the IMS change list.

IMSRSC repository change list processing encountered an error. The change list contains the names of resources and descriptors that were imported while the IMS was down.

If any of the resources or descriptors in the change list fails import, the import is stopped and none of the resource or descriptor definitions from the IMS change list are imported.

The resources and descriptors remain as NOTINIT until IMS is restarted and the next internal import from change list succeeds or the user runs an **IMPORT DEFN SOURCE(REPO)** command to successfully import the resource and descriptor definitions from the IMSRSC repository.

IMPORT processing error

The **QUERY DB, QUERY PGM, QUERY RTCODE**, and **QUERY TRAN** commands can be run to display the resources that are in the NOTINIT-xx-REPOCHGLIST state because of the internal import from the IMS change list.

There are no **QUERY** commands for displaying NOTINIT descriptors because descriptors do not include a status; therefore, look for message DFS4414I for the descriptor names that are still NOTINIT if the internal import from change list failed.

Example 2-1 shows the format of a DFS4401E message.

Example 2-1 Format of a DFS4401E message

```
DFS4401E RM requestname REQUEST FAILED, RC=rc RSN=rsn ERRORTEXT=errortext
```

RM return/reason codes indicate why import processing failed and are documented in CSLRRR macro, as shown in the following examples:

- RM reason code 5508 = Repository server not available
- RM reason code 5518 = Repository not available

DELETE DEFN command changes

IMS recommendation is to delete the IMS runtime definition from IMS by using the **DELETE** command followed by deleting the resource definitions from the IMSRSC repository by using the **DELETE DEFN** command.

However, if the recommended procedure is not followed and resource definition is deleted from the IMSRSC repository for the IMS before it is deleted in the IMS system, it still exists in the IMS after restart. In this case, the user must issue the **DELETE** command to delete the runtime resource definition after IMS restarts.

The IMS change list in the IMSRSC repository is not created when a **DELETE DEFN** command is issued and one or more IMS systems that are specified on the FOR() keyword are down.

As a result of the DELETE DEFN command with keyword DB, DBDESC, RTC, RTCDESC, PGM, PGMDESC, TRAN, or TRANDESC, definitions of the specified resource names are deleted from the IMSRSC repository for the specified IMS systems. Also, the DELETE DEFN command deletes the resource names from an IMS change list in the repository if a change list exists for one or more IMS systems that are specified on the FOR() keyword. Because the resource definition is deleted and is not available to be imported when the IMS system restarts, it is also deleted from the IMS system’s change list.

DELETE DEFN new TYPE option CHANGELIST

Any change list that is created for IMS when it is down when the **IMPORT** command is run is automatically deleted during the IMS warm, cold, or emergency restart. Any errors deleting the IMS change list leaves a residual change list. You can use the **DELETE DEFN TYPE(CHGLIST)** command to delete any residual IMS change list that cannot be deleted by IMS during IMS change list processing. If this procedure is not followed, any resource definition in the IMS change list is imported at the next restart.

IMS recommendation is to delete the IMS runtime definition from IMS by using the **DELETE** command followed by deleting the resource definitions from the IMSRSC repository by using the **DELETE DEFN** command.

However, if the recommended procedure is not followed and the resource definition is deleted from the IMSRSC repository for the IMS before it is deleted in the IMS system, it still exists in the IMS after restart. In this case, the user must run the **DELETE** command to delete the runtime resource definition after IMS restarts.

Error scenarios during IMS change list use

If the IMS system goes down without deleting the change list, another change list cannot be created at the next **IMPORT DEFN SOURCE(REPO) SCOPE(ALL)** command because the change list is locked and not accessible. You can use the **DELETE DEFN TYPE(CHGLIST)** to resolve the following situations:

- After a DFS4409A message (REPOSITORY CHANGE LIST IS NOT ACCESSIBLE) is issued and you issue an **IMPORT DEFN SOURCE(REPO) SCOPE(ALL)** command manually to successfully synchronize IMS with the repository.

  Because the IMPORT command is issued manually, the change list is not needed and should be deleted. If the change list is not deleted, it is processed at the next IMS restart.

- After a DFS4401E message is issued (RM CSLRPDEL REQUEST FAILED) when IMS cannot delete the change list.

  Because IMS cannot delete the change list automatically, you can use the **DELETE DEFN** command to delete the change list. If this residual change list is not deleted, it is deleted at the next IMS restart.

Example 2-2 shows the format of a DFS4401E message.

Example 2-2 Format of a DFS4401E message

```
DFS4411E REPOSITORY CHANGE LIST PROCESSING FAILED RC=rc RSN=rsn
```

IMS return/reason codes indicate why IMS change list processing failed and are documented in DFSCMDRR macro, as shown in the following examples:

- IMS reason code 4104 = No RM address space
- IMS reason code 4108 = No SCI address space

XRF configuration

In an XRF configuration, the recommendation is to issue the **EXPORT DEFN** command with the IMSid of the active and the alternative so the resource definitions are added to the IMS resource lists of the active and alternative system.

When an **IMPORT DEFN** command is issued on the XRF configuration, the IMS change list is created for the alternative system in the repository.

The IMS change list for the XRF alternative is not reprocessed at IMS takeover because all of the information in the IMS change list is processed as a part of the X’22’ and the checkpoint log records.

The IMS change list for the XRF alternative is deleted at end of takeover.

2.1.8 IMSRSC Repository contents

The IMS repository data sets store IMS resource definitions. There is one type of IMS repository; that is, the IMSRSC repository for managing DRD definitions.

An IMSRSC repository contains the following three major types of information:

- Resource lists

  These resource lists contain a list of all the programs, transactions, databases, and FP routing codes that are defined in a particular IMS system.

- Actual stored resource definitions

  The actual stored resource definitions for all programs, transactions, databases, and FP routing code definitions that are managed by the repository server.

  Different attribute values for a particular resource definition are supported by a generic definition and IMS-specific definitions where the attribute is different. An example is for the MSC SIDR and SIDL settings.

  > **Note:** Manual processes are required with Resource Definition Data Sets (RDDS) to maintain different attribute values.

- IMS change lists

  IMS change lists are new in IMS 13 and contain resource and descriptor names that correspond to resource changes that were made when an IMS was down when an IMPORT SCOPE(ALL) is issued.

  Behind the scenes, a separate IMS change list is created for each resource and descriptor type.

  If a change list exists for the IMS that is restarted, the database, program, transaction, and routing code resources and descriptors in the IMS change list that apply to the IMS environment are quiesced and are not available for use until the stored resource definitions are imported from the repository.

  They are applied at the next warm or emergency restart after the IMS log is processed via an internal import process for that particular IMS system that was down.

  For the resources or descriptors that are in the IMS change list and that do not exist in IMS, the runtime resource definitions are created from the stored resource definitions in the repository.

  For the resource or descriptors that exist in IMS, the runtime resource definitions are updated with the stored resource definitions from the repository.

  The change list for the IMS system is deleted at the end of the cold, warm, or emergency restart, which allows synchronization with the other IMS systems in the IMSplex.

  With RDDSs, manual coordination is required to handle IMS systems that are down.

IMS change lists in an XRF configuration

The IMS change list is created for an XRF alternative IMS during the **IMPORT DEFN SOURCE(REPO) SCOPE(ALL)** command if the resource definition is defined for the XRF alternative system.

When the XRF alternative is in restart mode, it is considered inactive. The IMS change list that is created for the XRF alternative is deleted during takeover without being processed because the resource definitions are created or updated on the alternative from the log records that are sent from the IMS active system.

If one exists, the IMS change list for the IMS active system is processed during the restart of the IMS active system, and the X'22' log record that was written during the internal import is processed on the XRF alternative to obtain the resource definitions.

The IMS change list is ignored and deleted during takeover.

The IMS change list for the XRF alternative is not reprocessed at IMS takeover because all the information in the change list is processed as a part of the X'22' log record and the checkpoint log records.

The IMS change list for the XRF alternative is deleted at the end of takeover.

2.1.9 IMPORT command with SCOPE(ACTIVE)

When you issue the **IMPORT DEFN SOURCE(REPO)** command, you can specify whether the import is applicable to the following components:

- All IMS systems in which the resource is defined (even if some of the IMS systems are not active), or
- All active IMS systems in the IMSplex

The keywords SCOPE(ALL) and SCOPE(ACTIVE) for the IMPORT DEFN command identify whether the command applies to all IMS systems in which the resource is defined or only to the active systems.

The SCOPE(ACTIVE) keyword applies the import to the active IMS systems in the IMSplex only. Any inactive IMS system does not complete the import because an IMS change list is not built for it in this case.

Therefore, when it warm starts or emergency restarts, its definitions are not synchronized with the definitions of the other IMS systems in the IMSplex.

To re-establish synchronization, you can run the **IMPORT** command to import the resources that the other active IMS systems imported while it was inactive.

If the inactive IMS is cold started, it is synchronized with the other IMS systems because it reads its entire IMS resource list.

Figure 2-4 on page 23 shows the use of the **IMPORT** command with SCOPE of ACTIVE.

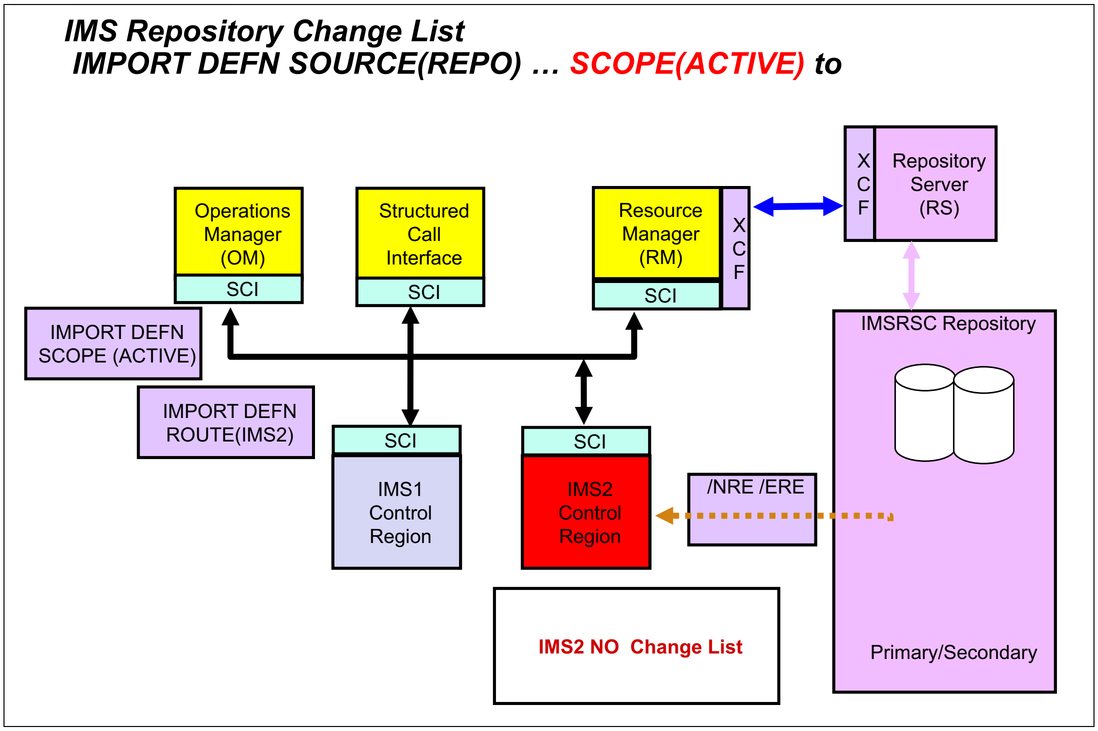


Figure 2-4 Use of the IMPORT command with SCOPE of ACTIVE

IMS2 was not active when the *IMPORT* command was entered and a change list for IMS2 is not created.

When IMS2 is started via normal or emergency restart, changes are not automatically imported.

After IMS2 completes initialization processing, the **`IMPORT DEFN SOURCE(REPO) ROUTE(IMS2)`** command must be entered for IMS2 and any other IMS systems that were inactive to synchronize runtime resource definitions in all IMS systems in the IMSplex.

To re-establish synchronization with the resource and descriptor definitions, you can enter an **`IMPORT`** command manually to import the resources that the other active IMS systems in the IMSplex imported while it was inactive.

To ensure resource and descriptor definition synchronization across all IMS systems, complete one of the following tasks:

- Cold start the inactive IMS

  If the inactive IMS is cold started, it is synchronized with the other IMS systems because it reads its entire IMS resource list.

  All of the resource definitions that are defined for the IMS are automatically imported from the repository.

- Warm start or emergency restart the inactive IMS

  A way to achieve synchronization is to warm start or emergency restart and then issue an **`IMPORT`** command to read in the stored definitions that might be updated while it was inactive.

  Use the **`ROUTE`** command to previously inactive IMS.

- Use IMPORT with SCOPE(ALL)

  Another way to maintain synchronization among the runtime definitions in the IMSplex is to run the **IMPORT** command with the recommended SCOPE(ALL) parameter.

2.1.10 Summary

Runtime definition changes can now be applied to an IMS when it warm starts or emergency restarts.

Changes made to active systems with the **IMPORT DEFN SCOPE(ALL)** command are applied to an inactive IMS when it is restarted.

The enhancements have the following benefits:

- Simplification of IMSplex management.
- Easier to maintain synchronization of IMS systems resources in a DRD with repository environment.

2.2 IMS user exit enhancements

The user exit enhancement provides the means to reduce the number of times IMS must be restarted to update a user exit by extending the existing refresh support to more user exits.

Consider the following points:

- The existing refresh support was expanded to include more existing user exits.
- These user exits are now referenced by their user exit type instead of the old default user exit routine.

2.2.1 Overview of REFRESH USEREXIT command as delivered in IMS 12

The **REFRESH USEREXIT** command was delivered in IMS 12 as an SPE under APAR PM56010 (PTF UK79071). By using this command, you can dynamically refresh supported user exit types while IMS is active and can be used to add, update, or delete user exits.

All exit types that support the enhanced user exit services are eligible for REFRESH, and are described in this section.

The following command refreshes the user exit types that are specified without bringing IMS down:

```
REFRESH USEREXIT TYPE(exittype)
```

The following exit types can be refreshed:

- ICQSEVNT
- ICQSSTEV
- INITTERM
- PPUE
- RESTART

The USER_EXITS section of the DFSDFxxx member is read and the exit routines that are listed for the type being refreshed are loaded.

If there are no errors that are encountered while the user exit type is processed, the new user exit routines replace the current routines and all subsequent calls to the user exit call the new routines.

The old routines are deleted when there are no more callers in any of the exit routines that are defined for the user exit type.

The optional MEMBER() parameter provides a means to point to a different DFSDFxxx member for testing.

If the parameter is used in production, the user must ensure that the default member is updated to ensure that the correct user exit routines are loaded if IMS must be restarted.

2.2.2 User exit dynamic refresh capability with IMS 13

The refresh of each exit type is independent. If multiple user exit types are refreshed with the same command, a failure for a particular user exit type does not affect the other user exit types that are part of the refresh. The refresh of the other user exit types can still be successful.

In addition to refreshing an exit with an updated version, you can use the **REFRESH USEREXIT** command to add or delete a user exit type to IMS.

To add a user exit type, insert EXITDEF statements into the USER_EXITS section of the DFSDFxxx member for the user exit type you want to add and then run the **REFRESH USEREXIT** command for that exit type.

To delete a user exit type, remove the EXITDEF statements from the USER_EXITS section of the DFSDFxxx member for the user exit type you want to delete and then run the **REFRESH USEREXIT** command for that exit type.

2.2.3 REFRESH USEREXIT example

In this section, we describe the USER_EXITS section and the Command response lines.

USER_EXITS section

Figure 2-5 on page 26 shows the USER_EXITS section of the DFSDFxxx member that is used for the command example. It contains more exit definitions, but they are not relevant to this command.

REFRESH USEREXIT Example

- Edit DFSDFxxx before issuing REFRESH USEREXIT command

```
<SECTION=USER_EXITS>
EXITDEF=(TYPE=ICQSSTEV,                          /* IMS CQS STR EVENT  */
         EXITS=(DFSCSTX2,DFSCSTX0,DFSCSTX1))    /* EXIT LIST          */
EXITDEF=(TYPE=ICQSEVNT,                          /* IMS CQS EVENT      */
         EXITS=(DFSCQEX1,DFSCQEX2,DFSCQEX0))    /* EXIT LIST          */
EXITDEF=(TYPE=INITTERM,                          /* IMS Init/Term Exit */
         EXITS=(DFSITRX2,DFSITRX3))              /* EXIT LIST          */
```

Modules DFSCSTX2 and DFSITRX3 are not in the load library

Figure 2-5 USER_EXITS section of the DFSDFxxx member

> **Note:** Two of the module names are not present in the load library, which ultimately prevents the ICQSSTEV and INITTERM exit types from successfully being refreshed.

Command response lines

Screen 1 of output

Figure 2-6 shows the command response lines for the command example. The exit types that are associated with the module names that are not present in the load library cannot be refreshed, which is reflected with a completion code (CC) of 92.

REFRESH USEREXIT Example – Screen 1 of Output

Command input: `REFRESH USEREXIT TYPE(ICQSTEV,ICQSEVNT,INITTERM)`

```
PLEX1                    IMS Single Point of Control
Command ===>
No clients returned return code 0. Check return code(s).
 -----------------------    Plex . .        Route . .            Wait . .
Response for: REFRESH USEREXIT TYPE(ICQSSTEV,ICQSEVNT,INITTERM...
ExitType ModName  MbrName   CC CCText
ICQSSTEV          SYS3      92 COMMAND PROCESSING ERROR
INITTERM          SYS3      92 COMMAND PROCESSING ERROR
ICQSEVNT DFSCQEX0 SYS3       0
ICQSEVNT DFSCQEX1 SYS3       0
ICQSEVNT DFSCQEX2 SYS3       0
```

Figure 2-6 Screen 1 of output

The modules that are of the ICQSEVNT exit type are successfully refreshed and the returned CC is 0. To see more information that is associated with the command response, press PF4 to view the log.

Screen 2 of output

Figure 2-7 shows the log after PF4 is pressed. More information is returned, but to see the most useful information, scroll to the bottom of the log data by pressing PF8.

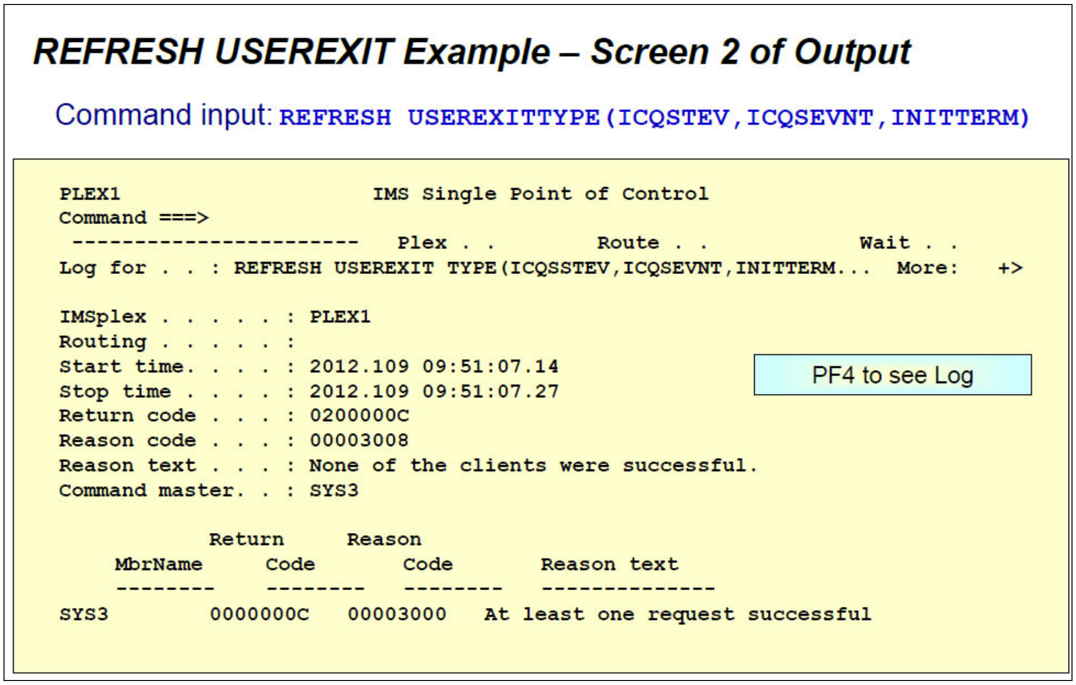


Figure 2-7 Screen 2 of output

Screen 3 of output

Figure 2-8 shows more details of the command response that was shown by pressing PF8 from the screen that is shown in Figure 2-7 on page 27.

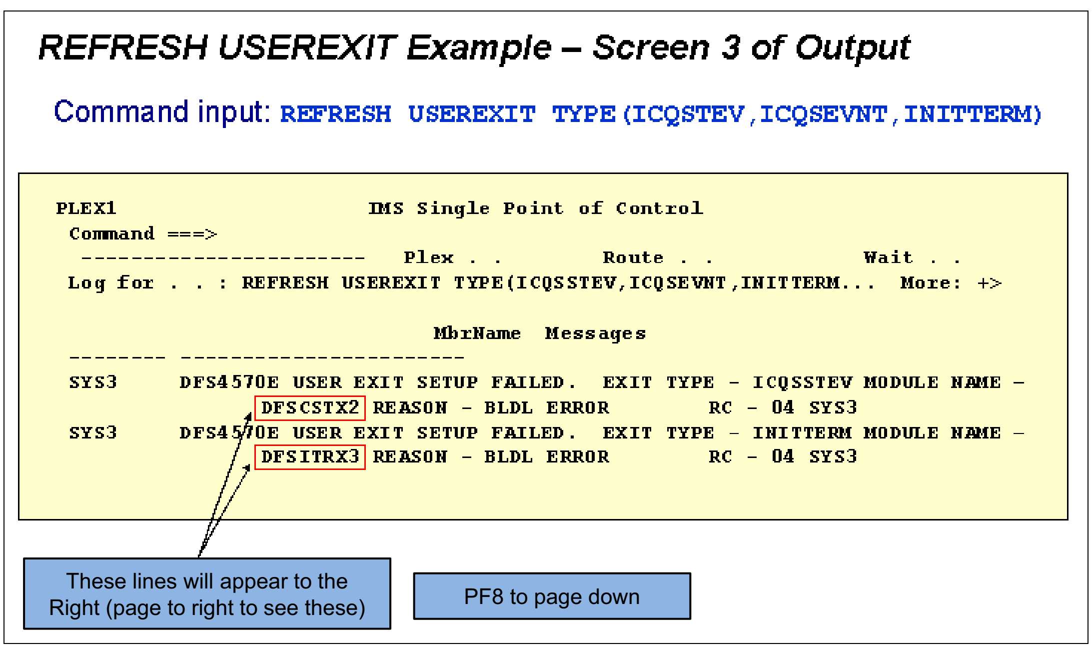


Figure 2-8 Screen 3 of output

Note: The two exit routine names that were missing from the load library are shown.

The other lines of output are scrolled off to the right, which is indicated by the `More: +>` text that is shown on the right side of the screen in Figure 2-8 on page 27.

2.2.4 IMS 13 user exit enhancements

IMS 13 adds enhanced user exit services support for more user exits. Users can now dynamically refresh more IMS user exit routines to bring in an updated version of the exit or add or delete it.

This functionality significantly reduces downtime because the IMS control region no longer requires restart.

This support not only includes the ability to dynamically add, update, or delete a user exit without restarting the IMS control region, but to display useful exit information and enable IMS to call multiple user exit routines from a single exit point.

Users can now display information about more user exits that are defined in the USER_EXITS section of DFSDFxxx, which provides useful exit information to the user.

Enabling more exits to use the enhanced user exit services expands flexibility, improves the availability of the IMS online environment, and makes it easier to manage user exits.

Users can now code an exit to use the ability of IMS to call multiple routines of the same type from a single exit point.

Benefits

This IMS 13 enhancement provides the following benefits:

- Expanded flexibility
- Increased IMS availability
- Simplified user exit management

The software and hardware requirements for this enhancement are the same as IMS 13.

Enhanced User Exit Services added to more exit types

Table 2-2 lists the user exits that can now use the enhanced user exit services in IMS 13.

Table 2-2 User exits that use the enhanced services in IMS 13

Function	Name	Description
BSEX	DFSBSEX0	Build Security Environment Exit
LOGEDIT	DFSFLGE0	Log Edit Exit
LOGWRT	DFSFLGX0	Logger Exit
NDMX	DFSNDMX0	Non-discardable Message Exit
OTMAIOED	DFSYIOE0	OTMA Input/Output Edit Exit
OTMARTUX	DFSYRTUX	OTMA Resume TPIPE Security Exit
OTMAYPRX	DFSYPRX0	OTMA Destination Resolution Exit
RASE	DFSRAS00	Resource Access Security Exit

The IMS 11 enhanced user exit services

The enhanced user exit services that now apply to a wider scope of user exits enable an exit to be dynamically added, changed, or deleted by using a **USER REFRESH** command. Its information is dynamically displayed by using a **QUERY USEREXIT** command.

These services also allow multiple exits of the same type to be called and run in the order that they are specified in the EXITDEF statement of DFSDFxxx.

The exits that are listed can now use enhanced user exit services to perform the following tasks:

- Use REFRESH USEREXIT to bring in new copy of an exit that is based on type:
  - Either all exits of a certain type as listed in DFSDFxxx are refreshed or none
  - Can also add or delete if DFSDFxxx updated first
- Use QUERY USEREXIT to display useful exit information
- Code multiple exits of the same exit type to be called from a single exit point

Considerations

If a user exit is called while it is being refreshed with a new copy, they each share the static work area.

During this time, it is possible that each exit accesses the work area and the new exit that is brought in with the **REFRESH USEREXIT** command must not reformat the work area.

An SXPL flag that is representing whether the work area is shared is set until the original exit is deleted. The SXPL flag is set when the exits are called. If the static work area is shared, the flag is set. If it is not shared, the flag is not set.

The original exit routines are deleted when there are no more callers in the exit routines and the SXPL flag is no longer set. It is at this point that the new, refreshed, version of the exit can proceed with reformatting the static work area.

For example, if a new user exit reformats the static work area, it should wait until this flag is reset because the original version of the exit might be relying on the work area to remain intact or unmodified.

Tip: Removing DFSRAS00 from DFSDFxxx then running the **REFRESH USEREXIT** command while also specifying ISIS=C/A in DFSPBxxx results in a message issued that indicates the exit is no longer called.

2.2.5 Command examples

In this section, we describe command examples for **REFRESH USEREXIT** and **QUERY USEREXIT**, including a scenario in which the dynamic refresh succeeds for some exit types and fails for others.

Although the command fails for some exit types, it succeeds for other types.

Adding and updating exits

Figure 2-9 on page 30 shows exits are successfully added or updated by running the following refresh command:

```
REFRESH USEREXIT TYPE(RASE,BSEX)
```

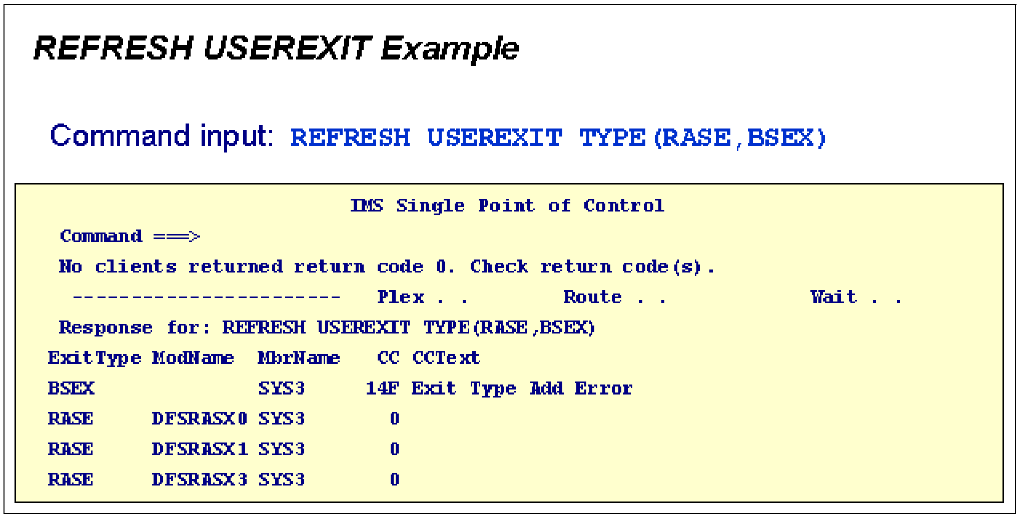


Figure 2-9 Adding and updating exits

Figure 2-10 shows one exit's refresh attempt that results in an error because of the lack of an EXITDEF statement in DFSDFxxx of the BSEX type.

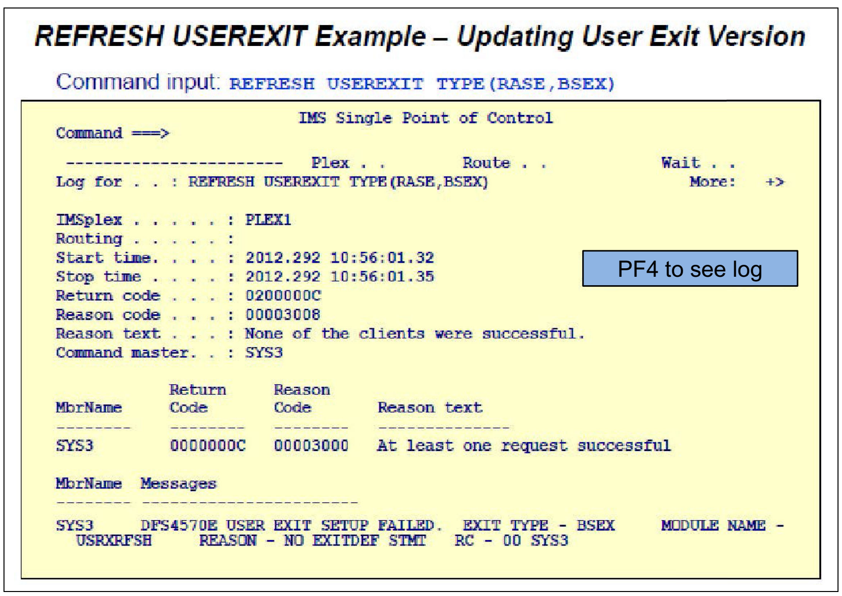
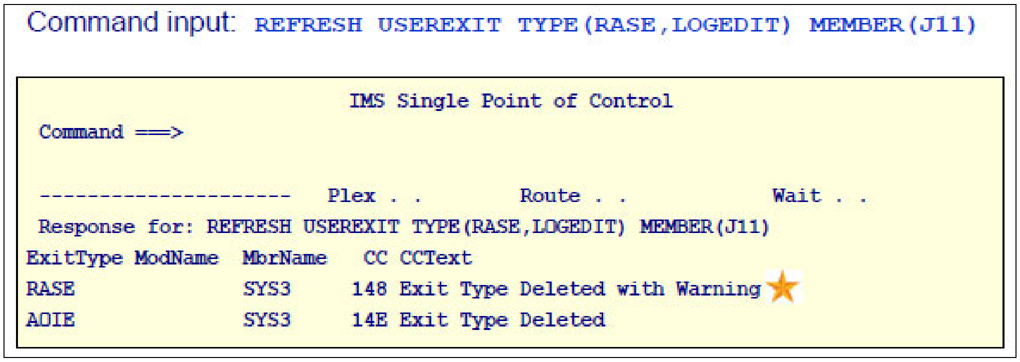


Figure 2-10 Adding and updating exits: Exits refresh error

Deleting exits

Figure 2-11 on page 31 shows exits are deleted by running the following refresh command:

```
REFRESH USEREXIT TYPE(RASE,LOGEDIT) MEMBER(J11)
```



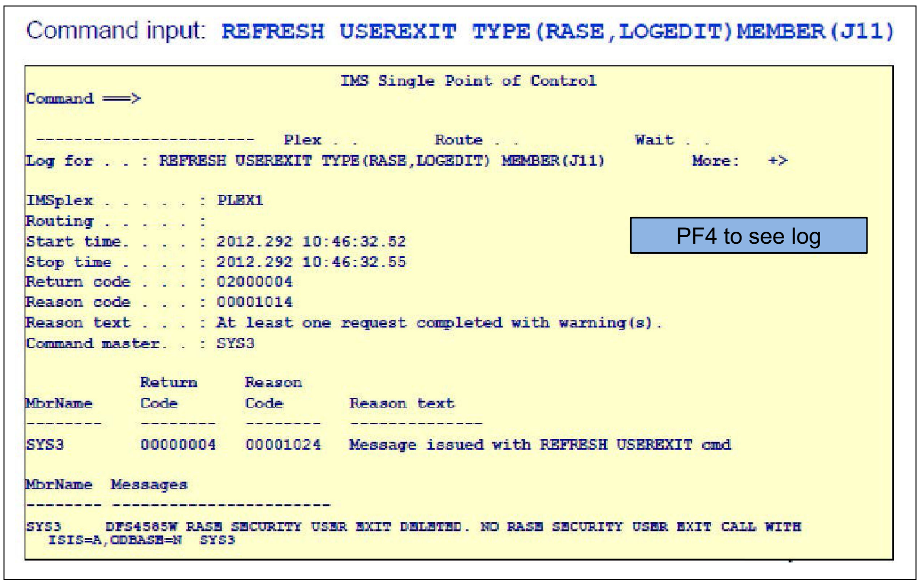
```
Command input: REFRESH USEREXIT TYPE(RASE,LOGEDIT) MEMBER(J11)

                         IMS Single Point of Control
 Command ===>

 ---------------------  Plex . .        Route . .            Wait . .
 Response for: REFRESH USEREXIT TYPE(RASE,LOGEDIT) MEMBER(J11)
ExitType ModName  MbrName   CC CCText
RASE              SYS3     148 Exit Type Deleted with Warning
AOIE              SYS3     14E Exit Type Deleted
```

Figure 2-11   Deleting a user exit

Figure 2-12 shows a refresh warning when exits are deleted.



```
Command input: REFRESH USEREXIT TYPE(RASE,LOGEDIT)MEMBER(J11)

                              IMS Single Point of Control
Command ===>

 ----------------------  Plex . .        Route . .            Wait . .
Log for . . : REFRESH USEREXIT TYPE(RASE,LOGEDIT) MEMBER(J11)        More:   +>

IMSplex . . . . . : PLEX1
Routing . . . . . :
Start time. . . . : 2012.292 10:46:32.52
Stop time . . . . : 2012.292 10:46:32.55
Return code . . . : 02000004
Reason code . . . : 00001014
Reason text . . . : At least one request completed with warning(s).
Command master. . : SYS3

            Return     Reason
MbrName     Code       Code       Reason text
--------    --------   --------   --------------
SYS3        00000004   00001024   Message issued with REFRESH USEREXIT cmd

MbrName  Messages
-------- ----------------------
SYS3     DFS4585W RASE SECURITY USER EXIT DELETED. NO RASE SECURITY USER EXIT CALL WITH
  ISIS=A,ODBASE=N  SYS3
```


Figure 2-12   Deleting a user exit: Exits refresh error

Query from user exits

Figure 2-13 on page 32 shows a query from several active user exit routines where one is not found by running the following query command:

```
QUERY USEREXIT TYPE(RASE,BSEX,NDMX,LOGEDIT)
```

```
Command input: QUERY USEREXIT TYPE(RASE,BSEX,NDMX,LOGEDIT)

                      IMS Single Point of Control
 Command ===>
 No clients returned return code 0. Check return code(s).
  -----------------------  Plex . .     Route . .          Wait . .
 Response for: QRY USEREXIT TYPE(RASE,BSEX,NDMX,LOGEDIT)
ExitType ModName  MbrName   CC CCText
BSEX              SYS3      10 NO RESOURCES FOUND
LOGEDIT  DFSLGEX0 SYS3       0
LOGEDIT  DFSLGEX1 SYS3       0
LOGEDIT  DFSLGEX2 SYS3       0
NDMX     DFSNDMX0 SYS3       0
NDMX     DFSNDMX1 SYS3       0
NDMX     DFSNDMX3 SYS3       0
RASE     DFSRASX0 SYS3       0
RASE     DFSRASX1 SYS3       0
RASE     DFSRASX3 SYS3       0
```

Figure 2-13 Query from user exits

2.2.6 Security considerations

To restrict access to the type-2 commands **REFRESH USEREXIT** and **QUERY USEREXIT**, define resource profiles in the OPERCMDS class and grant the required user access authorities.

Table 2-3 lists the Resource Access Control Facility® (RACF) definitions to restrict this access.

Table 2-3 RACF definitions to restrict the access

IMS Command	Command Keyword	RACF access authority	Resource name
REFRESH	USEREXIT	UPDATE	IMS.plxname.REFRESH.USEREXIT
QUERY	USEREXIT	READ	IMS.plxname.QRY.USEREXIT

2.2.7 Summary

The IMS 13 user exit enhancements enable more user exits for the following enhanced user exit services:

- Dynamic refresh
- Display information
- Call multiple exits of same type from single point

This enhancement has the following benefits:

- Expanded flexibility
- IMS availability is increased
- Management of user exits eased

2.3 IMS SECURITY macro removal

The enhancements to security in IMS 13 remove the support for the SECURITY stage-1 system definition macro. This configuration reduces the dependency of IMS security on system generation.

2.3.1 Highlights

Most security options that were previously specified by using the SECURITY macro are now specified by using the following initialization parameters:

- ISIS
- RCLASS
- RCF
- SECCNT
- SGN
- TRN

All IMS security settings can now be defined as IMS start parameters.

Updates to SECURITY macro in system definition (SYSGEN) are no longer required because of its removal. Previously, certain settings were defined only in SECURITY macro.

With the removal of the SECURITY macro, you no longer must specify the use of the Signon/off Security exit routine (DFSCSGN0) and the Transaction Authorization exit routine (DFSCTRN0) during system definition or system start.

Instead, if the exit routines are linked in one of the STEPLIB or LINKLIST libraries, IMS loads the exit routine.

Also, the following security exit routines no longer need to be bound to the IMS nucleus. They can run in 31-bit storage and can share a work storage area by using a standard technique:

- Signon/off Security exit routine (DFSCSGN0)
- Transaction Authorization exit routine (DFSCTRN0)
- Security Reverification exit routine (DFSCTSE0)

2.3.2 Benefits

The removal of SECURITY macro has the following benefits:

- A significant reduction in system programmers time and effort that is required in maintaining IMS systems.
- The management of security definitions is easier.
- The system definition (SYSGEN) process is made simpler.

The removal of selected user exits from IMS nucleus also has the following benefits:

- The exits are easier to maintain.
- The linking to IMS nucleus process is no longer necessary when one of the exits is changed.
- The usage of 24-bit storage is reduced.

2.3.3 New IMS start security parameters

You can now specify the RCLASS and SECCNT settings as IMS start parameters.

RCLASS parameter

The RCLASS parameter was added to the DFSPBxxx member, which co-exists with the ability to set this parameter within the DFSDCxxx member.

The support for the DFSDCxxx proclib member remains for users who specified it here instead of the SECURITY macro. If the RCLASS parameter is specified in both members, DFSPBxxx RCLASS parameter value overrides DFSDCxxx.

> **Note:** The RCLASS parameter specifies an identifier of 1 - 7 alphanumeric characters that are to be used to identity the IMS system as a resource class to RACF for transaction authorization and user ID verification.

SECCNT parameter

The SECCNT parameter was added to the DFSDCxxx member, which specifies the maximum number of terminal and password security violations to be accepted per physical terminal and the number of transaction command violations per transaction before the master terminal notification of such violations.

The default is 0, which nullifies notification to the master terminal. The number that is specified must be 0, 1, 2, or 3.

IMS Syntax Checker

The changes to DFSPBxxx and DFSDCxxx proclib members are supported by IMS Syntax Checker.

Retrofit SPE APARs/PTFs in IMS 11 and IMS 12

The following retrofit SPE APARs/PTFs are available for IMS start security parameter enhancement activation in IMS 11 and IMS 12:

- PM48203/UK74050 (IMS 11)
- PM48204/UK74051 (IMS 12)

If the RCLASS parameter is specified in DFSPBxxx/DFSDCxxx proclib members, you can also have the following APARs/PTFs applied to avoid an error message being issued:

- PM72199/UK82616 (IMS 11)
- PM73558/UK82617 (IMS 12)

2.3.4 SECURITY macro

The SECURITY macro in IMS SYSGEN (STAGE1) can no longer be used to specify IMS security options.

If the SECURITY macro remains in IMS 13 STAGE1, it results in a return code 2. Example 2-3 shows the error message that is issued.

Example 2-3 Error message due SECURITY macro specification

```
G115    SECURITY MACRO IS NOT SUPPORTED. SPECIFY SECURITY OPTIONS
        USING IMS EXECUTION PARAMETERS.
```

The SECURITY macro is ignored and IMS initialization continues.

The SECCNT parameter in the COMM and IMSGEN macro is still supported. Specifying SECCNT in the DFSDCxxx IMS PROCLIB member overrides the SECCNT specification in IMS SYSGEN.

2.3.5 User exits removed from IMS nucleus

The user exit DFSCSGN0 can be called at IMS initialization if it is included in STEPLIB and the user can now code the exit to obtain storage via STORAGE OBTAIN (or via the traditional GETMAIN macro) that can be shared with the DFSCTRN0 and DFSCTSE0 user exits by using a new parameter that contains the storage address.

The user exits DFSCSGN0, DFSCTRN0, and DFSCTSE0 are now linked separately. They are loaded from STEPLIB (if present) into 31-bit storage.

To use this new offering, the DFSCSGN0 must be updated to obtain the storage during IMS initialization and all three user exits must be updated to use this new parameter.

- The storage address is passed to IMS, then IMS passes the address to DFSCSGN0, DFSCTRN0, and DFSCTSE0. This configuration enables easier sharing of storage between these user exits.
- For more information about how to update the DFSCSGN0, DFSCTRN0, and DFSCTSE0 user exits to use the new parameter to share storage, see the exit interface documentation.

Operational considerations

The new DFS1937I message indicates which user exits were loaded and it can be used in automation to ensure that exits are being used.

You can change exits to be 31 bit and use the new parameter to share storage.

2.3.6 Communications between user exits

Because the DFSCTRN0, DFSCTSE0, and DFSCSGN0 user exits were removed from the nucleus in IMS 13, consideration must be given to maintaining their ability to communicate with one another. The following options are available:

- Treating the exits as stand-alone modules and the use of the *new parameter* to share the storage that is obtained during IMS initialization (recommended). Complete the following tasks:
  a. Update DFSCSGN0 to obtain storage during IMS initialization.
  b. Use the *new parameter* that contains the storage address to pass between the exits. This requires updates to each exit to use new parameter.
- Binding the user exit modules together by using ALIAS. No exit updates are required. Consider the following points:
  – If DFSCTSE0 is part of your DFSCTRN0 source, link DFSCTSE0 as an ALIAS of DFSCTRN0.
  – If V addresses are used to exchange data among the DFSCTRN0, DFSCTSE0, and DFSCSGN0 user exits, link them together by using ALIAS.

In either case, the modules should be bound as re-entrant and as AMODE/RMODE 31 to prevent them from being loaded multiple times. Consider the following points:

- A re-entrant module can be used by multiple callers simultaneously in which concurrent activity is taking place.
- It is written so that none of its code is modifiable (no values are changed) and it does not track anything.
- The callers track their own progress (variables, flags, and so on); therefore, one copy of the re-entrant routine can be shared by any number of callers.

> **Important:** Keep an original version of each user exit to enable easy fallback to IMS 11 or IMS 12 if it is needed.

Sample for binding user exit modules together

Figure 2-14 shows a JCL sample for maintaining communication ability between the DFSCTRN0, DFSCTSE0, and DFSCSGN0 user exits.

Modules were bound as re-entrant (RENT) with AMODE/RMODE 31 to prevent multiple loads.

```
//LINK1   EXEC PGM=IEWL,
//         PARM=(,
//             NCAL,XREF,LIST,RENT RMODE=ANY,AMODE=31)
//SYSPRINT DD SYSOUT=A
//TEXT      DD UNIT=SYSVIO,DISP=(OLD,PASS),DSN=&&IMSDC33
//SYSLMOD   DD DSN=IMSTESTL.TNUC0,DISP=SHR,UNIT=SYSDA,
//             VOL=SER=USER01
//RESLIB    DD DSN=IMSTESTL.TNUC1,DISP=SHR,UNIT=SYSDA,
//             VOL=SER=USER01
//          DD DSN=IMSBLD.I13RTSMM.ARESLIB,DISP=SHR
//SYSUT1    DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//SYSLIN    DD *
  INCLUDE   TEXT(DFSCTRN0)
  INCLUDE RESLIB(DFSCSGN0)
 ENTRY        DFSCTRN0
 ALIAS        DFSCTSE0
 ALIAS        DFSCSGN0
 NAME         DFSCTRN0(R)
```

Figure 2-14 Binding user exit modules

> **Note:** The three modules were linked by using ALIASing, as shown in the lower portion of the JCL with DFSCTSE0 in Figure 2-14 and DFSCSGN0 being linked as ALIASes of DFSCTRN0.

Benefits

All IMS security settings can now be defined as IMS start parameters through SECURITY macro being removed. Consider the following points:

- There is reduction in system programmers time and effort that is required in maintaining IMS systems.
- The management of security definitions is easier.
- The system definition (SYSGEN) process is simpler.

Move security user exits out of the IMS nucleus into 31-bit storage for the following reasons:

- These exits are easier to maintain.
- The linking process to IMS nucleus is no longer necessary when one of the exits is changed.
- The usage of 24-bit storage is reduced.

2.4 RACF password phrase support

After the IMS 13 APAR PM85849, PTFs UI13516, and UI13517 are applied, the **/SIGN** command supports submitting password phrases to RACF for authentication.

The **/SIGN** command is enhanced with new keywords and valid values to support setting, changing, and entering password phrases for RACF authentication.

Password phrases include security advantages over passwords in that they are long enough to withstand most hacking attempts, yet are unlikely to be written down because they are easy to remember.

As defined by RACF, a password phrase is a character string that consists of mixed-case letters, numbers, and special characters including blanks. Consider the following points:

- Password phrases can be up 9 - 100 bytes in length.
- If a password phrase is less than 9 bytes, IMS passes it to RACF as a password instead of as a password phrase.
- Use of a password phrase is optional because 8-byte passwords can continue to be used.

IMS also supports passwords phrases that are passed by VTAM® logon data; however, password phrases cannot be changed by using VTAM logon data.

The **/OPNDST** command does not support password phrases. The default MFS panels also do not support password phrases.

2.4.1 TM Resource Adapter

Users of TM Resource Adapter, IMS Connect, the **/SIGN** command, and VTAM can now sign on to IMS by using RACF password phrases (“passphrases”) that are a minimum of 9 bytes and a maximum of 100 bytes. The passphrase is sent to RACF at authentication time.

IMS TM Resource Adapter (TMRA) can now include a passphrase in a message that it builds and passes to IMS Connect. IMS Connect, in turn, is enhanced to accept this type of message from TMRA that includes a passphrase.

Also, users can now change their passwords by using the message that is sent from TMRA to IMS Connect by using the format that is specified in the message.

2.4.2 IMS Connect

IMS Connect can accept messages from IMS TM Resource Adapter that contain a passphrase.

Users can change their password phrase in the application data section of the message that is received from TMRA.

The application data section features the following format:

```
'oldphrase' 'newphrase' 'newphrase'
```

To preserve single quotation marks that are part of the passphrase, add another single quotation mark next to each one.

Figure 2-15 shows the layout of a TMRA message.

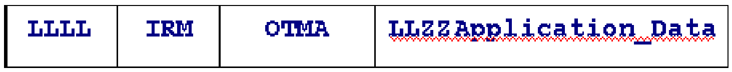


Figure 2-15 Layout of a TMRA message

2.4.3 Usage

IMS TMRA builds the IRM and the OTMA headers, which are sent to IMS Connect with each input message. IMS TMRA externalizes the ability to define the passphrase through its connection factory interface in a fashion similar to how users define user IDs and passwords.

For this new capability, IMS TMRA adds an IRM extension that contains the password phrase to the message header. When IMS Connect receives the request, the user ID and the new passphrase are used for authentication.

Figure 2-16 shows the fields of the TMRA message.

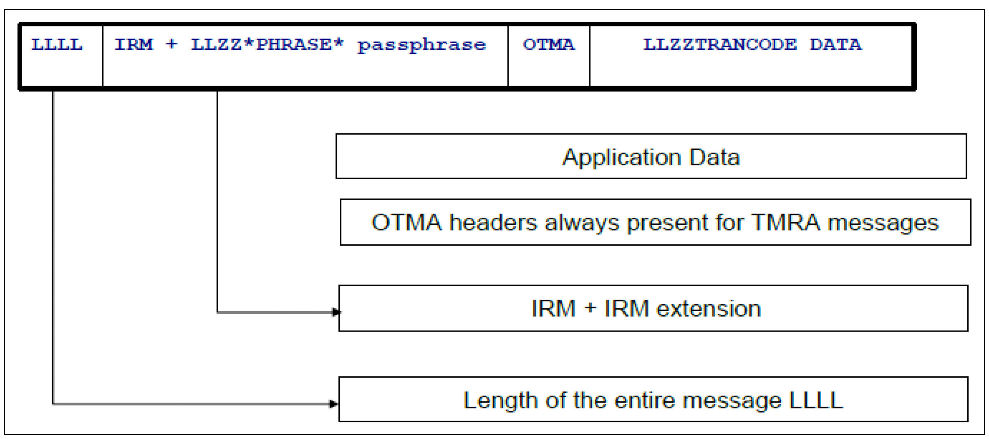


Figure 2-16 Fields of the TMRA message

The password change data is included in the Application Data section of the message that TMRA sends to IMS Connect.

The password change data comes after an eye catcher of "HWSPWCH". This is the default eye catcher of the HWSJAVA0 user message exit. The user can change it to any other value.

Figure 2-17 on page 39 shows the use of the password change format that was shown in Figure 2-15.

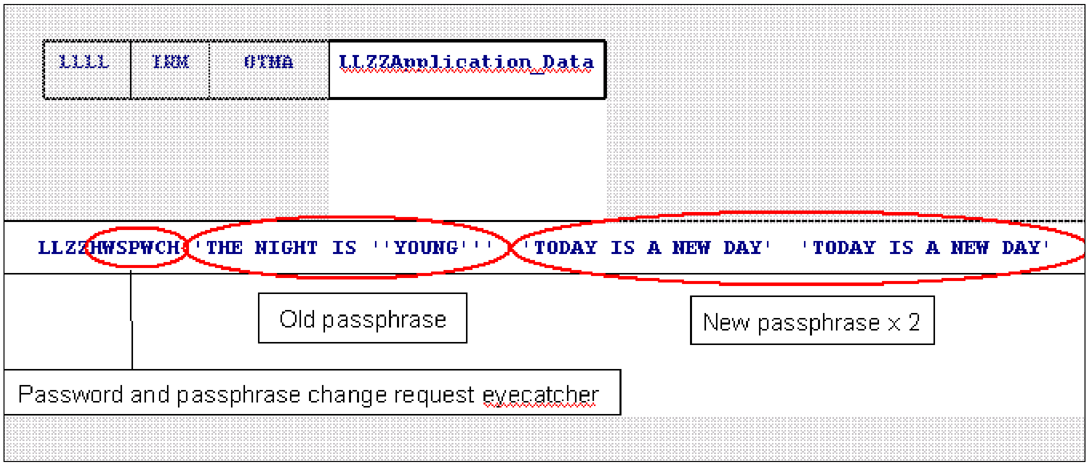


Figure 2-17 Password change format

2.4.4 MFS panels

MFS panels were updated to accommodate password phrases. Users can still enter passwords that are less than 9 bytes.

If an 8-byte password phrase is used on these panels, it is passed to RACF as a password.

2.4.5 IMS /SIGN command enhancement

The IMS /SIGN command was enhanced to support passphrases. Two new versions of this command are now available: **`/SIGN PASSPHRASE`** and **`/SIGN PASSPHRASEQ`**.

The traditional password support with **`/SIGN ON <userid> <userpw>`** is maintained.

Figure 2-18 shows the format of the new **`SIGN`** command that is enhanced to support passphrases.

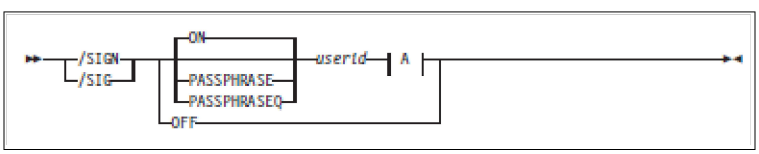


Figure 2-18 New SIGN command

/SIGN PASSPHRASE

The **`/SIGN PASSPHRASE`** option is used when a passphrase is specified and is most appropriate with an MFS panel because this command requires a passphrase that is 100 bytes.

This format can be ensured by an MFS panel in which the password phrase is entered. These password phrases can contain any character and are meant to be easier to remember than the possibly cryptic 8-character password.

IMS removes leading and trailing blanks, and if it is less than 9 bytes, IMS passes it as a password to RACF.

/SIGN PASSPHRASEQ

The use of `/SIGN PASSPHRASEQ` option is most appropriate when a user is entering log on credentials and is used when a passphrase is specified with single quotation marks. Consider the following points:

- IMS does not remove leading or trailing blanks.
- There is no requirement for the passphrase to be 100 bytes when this command is used and involves the use of single quotation marks.

When you are changing password phrases, you must specify another password phrase. This requirement is the same for passwords that you can change to another 8-byte password only.

VTAM logon user data

Log on user data that is passed to VTAM also now supports the use of passphrases. Example 2-4 shows the log on format.

Example 2-4 Log on format

```
LOGON APPLID(APPL1) DATA(‘USRT001 “this is my passphrase”’)
```

Note: Single and double quotation marks are used in this format. The entire DATA( ) parameter that contains the logon credentials is encapsulated in single quotation marks, and the passphrase is encapsulated in double quotation marks.

2.4.6 Changes to IMS user exits

If RACF password phrases are used, the new default DFSGMSG0 MFS panel that supports passphrases can be used. The DFSCSGN0 and DFSLGNX0 exits must be updated so that they can manage the passphrases that are passed to them.

Table 2-4 summarize the changes in the exits that must be updated.

Table 2-4 Exits needed to be updated

Name	Description	Action
DFSGMSG0	Greeting Messages user exit	Can be changed to the default MFS panel to use password phrase.
DFSCSGN0	Signon/off security user exit	Needs User Verification String (UVS) to change if password phrase is used.
DFSLGNX0	Logon user exit	Must accommodate log on data, including a password phrase that is passed to it.

2.4.7 Usage considerations

Passwords and passphrases are supported by IMS. Although the use of RACF passphrases is optional, the MTO, WTOR, and TCO terminal types all have limited space in terms of entering user ID and password or passphrase data. Because of this limitation, use passwords when you are signing on from these types of terminals.

Password phrases can be used for these terminal types. Because there is limited space, log on with /SIGN ONQ because it does not require the 100 bytes as does /SIGN ONP.

2.4.8 Summary

The RACF password phrases can now be used with the following interfaces:

- IMS TMRA
- IMS Connect
- **/SIGN** command
- VTAM logon user data

RACF password phrases include the following benefits:

- More robust
- Up to 100 bytes
- Can contain mixed-case letters, numbers, and special characters
- Easier to remember

2.5 DIAGNOSE command enhancements

The **/DIAGNOSE** command is enhanced to improve the reliability of diagnostic information and to streamline the problem determination process.

2.5.1 Highlights

The IMS **/DIAGNOSE** command is enhanced with new output options and includes information about more resource types.

Simplification for capturing diagnostic data

The process for capturing diagnostic data that is used in troubleshooting IMS issues is simplified and includes the following enhancements:

- SYSOUT option now available for /DIAGNOSE SNAP output:
  - Documentation can be gathered and stored in a readable format that is easy to retrieve.
  - Time-consuming SYSLOG searches and manual data formatting before transmission is no longer required.
- The **/DIAGNOSE SNAP** command was extended to include more resources and more coverage of existing resources:
  - A new SNAP resource type: MSNAME
  - Extended functionality for the following resource types: DB, AREA, MODULE, and BLOCK
  - Support for output filtering for the following resource types: LTERM, NODE, and USER

2.5.2 Benefits

The following benefits are included:

- Cost effective and nondisruptive alternative to console dumps.
- The **/DIAGNOSE** command is now more interactive and can be used more as a tool for easing the real-time diagnosis process.

- Decreased time and effort is required for capturing diagnostic information.
- Improved turn-around time in problem resolution.

2.5.3 SYSOUT output formatting routine

Users can now send formatted **/DIAGNOSE SNAP** command output to a SYSOUT data set, which enables easy submission to IBM support. Potential users include IMS application developers, DBAs, system programmers, support technicians, and developers.

Also, the new **OPTION(SYSOUT)** parameter was added to **/DIAGNOSE SNAP** command. SNAP data is queued to the diagnostic asynchronous work element (AWE) service and written to a SYSOUT data set.

SYSOUT data set includes documentation that is formatted and readable. No more manual copying or reformatting of MVS™ SYSLOG or cutting and pasting into a text file i required.

The following improvements also are included:

- Easy to retrieve. No more lengthy SYSLOG searches.
- New **/DIAGNOSE SNAP** command contains subparameters that manage output processing.

/DIAG SNAP…OPTIONS(SYSOUT) subparameters

You can specify the following keywords with the SYSOUT option:

- CLASS

  CLASS( ) specifies an output class for the SYSOUT data set. Any valid JES output class can be specified. Class can include the characters A - Z and 0 - 9. The default value of the output class is IMS control regions assigned class.

  Example 2-5 shows a **/DIAGNOSE SNAP** command with CLASS specified.

  Example 2-5 /DIAGNOSE SNAP command with CLASS specified

  ```
  /DIAGNOSE SNAP REGION(1) SHOW(ALL) OPTION(SYSOUT,CLASS(S))
  ```

- LIMIT

  LIMIT( ) specifies a limit for the number of lines of formatted SNAP data to process in response to the command. It is similar to OPTION(DISPLAY LIMIT( )). The valid range is 1 – 99999 with a default of 19999

- FORMAT

  FORMAT( ) specifies the format of the output to be produced. It also now applies to OPTION(DISPLAY) and includes the following types:

  – FORMAT(LONG)

    The default that produces a complete display that includes block name, description, location, and a dump of the complete storage area for the block in hexadecimal and character formats.

  – FORMAT(LOCATION)

    Produces block name, description, and location only

/DIAGNOSE SNAP command usage with OPTION(SYSOUT)

Figure 2-19 on page 43 shows the use of the **/DIAGNOSE SNAP** command, including OPTION(SYSOUT) messages and data set output.

```
/DIAG SNAP LTERM(SMASTER) OPTION(SYSOUT)
DFS3789I DIAGNOSE COMMAND SNAP LTERM QUEUED TO SYSOUT TKN(00000002) IVP1
DFS058I 08:31:30 DIAGNOSE COMMAND COMPLETED   IVP1
DFS3788I DIAGNOSE SYSOUT DATA SET STC.IMS.STC14202.D0000104.? OPENED
 FOR SNAP LTERM TKN(00000002) IVP1
DFS3788I DIAGNOSE SYSOUT DATA SET STC.IMS.STC14202.D0000104.? CLOSED
 FOR SNAP LTERM TKN(00000002) IVP1
```

Figure 2-19 Generated OPTION (SYSOUT) messages

Figure 2-20 shows the created data set output.

```
 /DIAGNOSE SNAP STORAGE DISPLAY

    Resource: LTERM(SMASTER)

    CNT      Communication Name Table (Trgt)  Loc: 25A2D418
    ---- -------- -------- -------- --------  ----------------
    0000 8004A5C4 00000000 00000000 00000002 |..vD............|
    0010 00000000 12820065 00000020 E2D4C1E2 |.....b......SMAS|
    0020 E3C5D940 40000001 0004B310 25A2D2B0 |TER  ........sK.|
    0030 00000000 00000000 22250120 00000000 |................|
    0040 25A2D490 00000000 0004A618 00000000 |.sM.......w.....|
    0050 00000000 00000000 00000000 00000000 |................|
    0060 00000000 00000000 00000020 00000000 |................|
    *14139/083130*
```

Figure 2-20 Created data set output

Benefits of support added

The value of reduced time and effort that is involved in diagnostic data capture process is because of the following factors:

- Formatting and coping and pasting into text file is no longer needed.
- Users no longer need to search SYSLOG for the wanted data.

The added support of DIAGNOSE SNAP also allows automated diagnostic data capture which are helpful for intermittent problems and facilitates historical tracking of a resource.

2.5.4 /DIAGNOSE SNAP ... SHOW( ) support added

IMS 13 adds support to the SHOW() parameters in the **/DIAGNOSE SNAP** command. The SHOW() parameters are not new in IMS 13. However, what is new is that they can now be used with a **/DIAG SNAP** command that captures information for a logical terminal, node or user, as shown in the following example:

```
/DIAGNOSE SNAP LTERM( ), NODE( ), USER( ) – SHOW( )
```

SHOW() parameter for logical terminals, nodes, and users

Users can now use the SHOW() parameter when diagnostic information is gathered for logical terminals, nodes, and users, as with other resource types.

Previously, there was no filtering mechanism available.

SHOW( ) parameters new in IMS 13

Table 2-5 lists the SHOW( ) parameters that are not new in IMS 13.

Table 2-5 SHOW( ) parameters not new in IMS 13

SHOW parameter	Function
SHOW(PRI)	Captures the primary control blocks for the resource and is the default.
SHOW(ALL)	Captures all control blocks available for the resource.
SHOW(OPT)	Captures all optional control blocks for the resource.
SHOW(blkname)	Captures the specified block by block name.
SHOW(blkname,blkname)	Captures multiple block names that are separated by commas.

There are differences between a primary and optional control block.

A *primary* control block represents a resource in that it is the primary or main block for the resource. Primary blocks are returned by a known and trusted source (such as **FIND DEST**) so we trust that the address and storage are valid (if the address or storage was bad, find dest fails first).

All other blocks are considered *optional*. Optional blocks are not trusted in that an optional block is never directly referenced. Instead, we use the copy routine and its ESTAE to copy the block to a known location safely and then reference fields in the block from the copy.

SHOW( ) parameters new in IMS 13

Table 2-6 lists the SHOW( ) parameters that are new in IMS 13 and apply only to capturing information for a logical terminal, node, or user.

Table 2-6 SHOW( ) parameters new in IMS 13

SHOW parameter	Function
SHOW(RECANY)	This parameter captures the RAQE and RAQERES blocks. It also displays the VTAM receive in or out buffer control blocks, and is originated from a customer requirement. This parameter can be used with LINE( ) and LINK( ).
SHOW(SA)	This parameter Is a synonym that can be used for capturing the save area set control block, and it is equivalent to SHOW(SAVEAREA). This parameter can also be used with LINE( ) and LINK( ).
SHOW(DEF)	This parameter captures the default, primary blocks for a logical terminal. It has the same functionality as SHOW(PRI). This parameter is for use with LTERMS only.
SHOW(TAR)	Captures the target CNT for a local logical terminal. This parameter is for use with LTERMS only.

Benefits of support added

The value of these enhancements enable users to have consistent control over the type and amount of data that is produced for resources that are involved in the **/DIAG SNAP** command.

With this improved level of control, the overall amount of time and effort that is required in capturing diagnostic data in analyzing and solving problems is reduced.

Potential users of these enhancements include IMS system programmers, support technicians, and developers.

Not only can the LTERM( ), NODE( ), and USER( ) resources now use the SHOW( ) filter with the **/DIAG** command to narrow the diagnostic data that is captured, but they can have more control blocks captured compared to previous IMS releases.

Example 2-6 shows a few examples of the **/DIAG SNAP** input command.

Example 2-6 Examples of the /DIAG SNAP input command

```
/DIAGNOSE SNAP LTERM(ltermname) SHOW(blockname)
/DIAGNOSE SNAP NODE(nodename) SHOW(blockname)
/DIAGNOSE SNAP USER(username) SHOW(blockname)
```

2.5.5 /DIAGNOSE SNAP ... RM( ) support added

Another way the **/DIAG** command was enhanced for the LTERM( ), NODE( ), and USER( ) resources is that it can now include the RM( ) parameter to specify the search scope for the resources that are involved in the command processing. A user can now choose to capture diagnostic data for a local resource only, a global resource only, or for both.

The following RM() parameters are available:

- RM(YES)

  The default. If the resource is not found locally, the RM resource structure is searched to find a global resource for use in the diagnostic data capture.

- RM(NO)

  Performs only a local search to find the resource.

- RM(ONLY)

  Performs only a global search on RM structure

In any scenario that RM is requested but is unavailable, the error message that is shown in Example 2-7 is issued.

Example 2-7 Error message if RM is requested but is unavailable

```
DFS2859I DIAGNOSE COMMAND FAILED - RM(ONLY) INVALID, RM UNAVAILABLE
```

Benefits of support added

The value of this new parameter allows users to more efficiently designate the search scope for the data that is captured, while providing increased flexibility.

Users can now exclusively capture diagnostic data for a local resource instead of a global resource and vice versa. Therefore, they can compare the local copy with the global copy to determine whether they match.

Because users can more efficiently capture the diagnostic data that they are specifically interested in, overall time and effort that is expended in problem analysis is reduced.

2.5.6 /DIAGNOSE SNAP BLOCK( ) command expanded

The scope of the **/DIAGNOSE SNAP BLOCK( )** command is expanded in that users can now include multiple single block instances.

Previously, only one single instance block was specified at a time (or ALL).

Benefits of command expansion

The value of this command expansion gives user more control over which blocks must be captured and ensures consistency in capturing diagnostic data. It also reduces overall time and effort in analyzing and solving problems.

Potential users include IMS system programmers, support technicians, and developers.

Example 2-8 shows an example of capturing information for the Log Control Directory control blocks (Journal and Monitor).

Example 2-8 Example of capturing information for the Log Control Directory control blocks

```
/DIAGNOSE SNAP BLOCK(LCD,LCDM)
```

2.5.7 /DIAGNOSE SNAP BLOCK( ) more blocks added

More blocks can now be specified as BLOCK( ) parameter keywords.

Table 2-7 lists the supported blocks (new blocks are highlighted in gray).

Table 2-7 Supported control blocks

Name	Block description
ALL	Captures information for all valid control blocks currently available.
CATA	Captures information for the Catalog Anchor Block control block. The CATA is available only in a DB/DC or DBCTL environment. If the CATA is requested in a DCCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DCCTL environment.
CMDE	Captures information for the Commands SCD Extension control block.
CSCD	Captures information for the APPC/OTMA SMQ SCD Extension control block. The CSCD is available only in a DB/DC or DCCTL environment. If the CSCD is requested in a DBCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
CSLA	Captures information for the Common Service Layer Anchor Block control block.
DCCB	Captures information for the Data Communications control block. The DCCB is available only in a DB/DC or DCCTL environment. If the DCCB is requested in a DBCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
DFA	Captures information for the Definition Anchor Block control block.
DGA	Captures information for the Diagnostics Anchor Block control block.
DGSD	Captures information for the Diagnostic Data Set Structures control block.
DGSW	Captures information for the Diagnose Work Area Storage control block.

Name	Block description
EDBT	Captures information for the RSR FP Global DB Tracking control block. The EDBT is available only in an IMS system where Fast Path is defined. If the EDBT is requested in a non-FP environment, a DFS154I error message is issued in response. This block is ignored when the ALL option is specified in a non-FP environment.
ESCD	Captures information for the Extended System Contents Directory control block. The ESCD is available only in an IMS system where Fast Path is defined. If the ESCD is requested in a non-FP environment, a DFS154I error message is issued in response. This block is ignored when the ALL option is specified in a non-FP environment.
GDBT	Captures information for the System RSR Global DB Tracking control block. The GDBT is available only in a DB/DC or DBCTL environment. If the GDBT is requested in a DCCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DCCTL environment.
LCD	Captures information for the Log Control Directory (Journal) control block.
LCDM	Captures information for the Log Control Directory (Monitor) control block.
LSCD	Captures information for the APPC SCD Extension control block. The LSCD is available only in a DB/DC or DCCTL environment. If the LSCD is requested in a DBCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
MWA	Captures information for the Modify Work Area control block.
QSCD	Captures information for the Queue Manager SCD Extension control block. The QSCD is available only in a DB/DC or DCCTL environment. If the QSCD is requested in a DBCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
RECA	Captures information for the VTAM Receive Any I/O Buffers control blocks. The RECA is available only in a DB/DC or DCCTL environment. If the RECA is requested in a DBCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DBCTL environment. Because of the volume of data that can be produced by snapping the VTAM Receive Any I/O Buffers, the RECA option is not included when the BLOCK(ALL) option is processed.
RSR	Captures information for the Remote Site Recovery Anchor control block.
SCD	Captures information for the System Contents Directory control block.
SDTT	Captures information for the Shutdown Trace Table control block.
SQM	Captures information for the Shared Queues Master Control Block control block. The SQM is available only in an IMS system where Shared Queues is defined. If the SQM is requested in a non-SQ environment, a DFS154I error message is issued in response. This block is ignored when the ALL option is specified in a non-SQ environment.
TIME	Captures information for the Timer Services SCD Extension control block.
TRA	Captures information for the Table Trace control block.
TSCD	Captures information for the OTMA SCD Extension control block. The TSCD is available only in a DB/DC or DCCTL environment. If the TSCD is requested in a DBCTL environment, a DFS110I error message is issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.

2.5.8 /DIAGNOSE SNAP DB( ) support for all blocks

Users can now gather diagnostic information about all control blocks that are associated with a specified database by using the `/DIAG SNAP DB()` command.

Previously, only the DDIR control block was captured.

The list of blocks that can be captured depends on the DB type. The following blocks are common to all DB types and snap blocks specific to a particular DB type:

- Full Function
- Fast Path

Table 2-8 lists the supported blocks that can be captured for a database. Consider the following points:

- Primary (P) control blocks are always present and available for capture.
- Optional (O) control blocks might be present and available for capture depending on workload, database type, and other factors.

Table 2-8 Supported control blocks that are captured for a database

Name	Block description	Macro	Avail
DDIR	Database Directory Block	DFSDDIR	P
DDIREXT	Database Directory Block Extension	DFSDDIR	O
RSCX	Resource Extension Block	DFSRSCX	O
DYNALMBR	Dynamic Allocate Member	DFSMDA	O
DBQLE	Database Quiesce List Entry	DFSDBQQL	O
EEQE	Database Quiesce List Entry	DFSEEQE	O
RRE	Residual Recovery Element	DFSRRE	O
TDBC	Tracking Data Base Control Block	DFSTDBC	O
SDTE	Segment Delta Block Table Entry	DFS5FLDD	O
DMB	Data Management Block	DFSDMB	O
PSDB	Physical Segment Descriptor Block	DFSDMB	O
SDB	Segment Descriptor Block	DFSSDBM	O
FDB	Field Descriptor Block	DFSFDB	O
DMBCPAC	Segment Edit/Compression Block	DFSDMB	O
DMBSEC	DMB Secondary List	DFSDMB	O
DMBDACS	Randomizer Control Block	DFSDMB	O
DMBAMPPR	Access Method Prefix Block Prefix	DFSDMB	O
DMBAMP	Access Method Prefix Block	DFSDMB	O
DCBACBP	Primary DCB/ACB Block	DCBD	O
DCBACBS	Overflow DCB/ACB Block	DCBD	O
DMBXBLCK	Data Management Exit Block	DFSDMB	O
DMBXARRY	Exit Array Entry Block	DFSDMB	O
DMBXT	Exit Description Block	DFSDMB	O
DMCB	DEDB Master Control Block	DBFDMCB	O
BHDR	Main Storage Database Header	DBFBMSDB	O

This enhancement ensures consistency and simplifies data capture process. It also requires less overall time and effort in analyzing and solving problems.

Potential users include IMS application developers, DBAs, system programmers, support technicians, and developers.

Example 2-9 shows an example of this enhanced command.

Example 2-9 Example of /DIAGNOSE SNAP DB( ) enhanced command

```
/DIAGNOSE SNAP DB(dbname) SHOW(ALL)
```

Example showing Full Function DB( ) control blocks

Figure 2-21 shows the output from the following **/DIAG** command against a Full Function DB:

```
/DIA SNAP DB(AUTODB) SHOW(DDIR,RSCX) OPTION(SYSOUT)
```

```
/DIAGNOSE SNAP STORAGE DISPLAY

   Resource: DB(AUTODB)

   DDIR     Database Directory Block         Loc: 22286040
   ---- -------- -------- -------- --------  ----------------
   0000 00000000 00000000 C1E4E3D6 C4C24040 |........AUTODB  |
   0010 00000000 00020400 001C001C 00000001 |................|
   0020 00000000 00000000 03000000 00000000 |................|
   0030 22286108 00000000 00000000 C4000000 |../.........D...|
   0040 00000000 00000000 00000000 00000000 |................|
   0050 00000000 00000000 00000000 00000000 |................|
   0060 00000000 00000000 00000000 00000000 |................|
   0070 00000000 00000000 25A51B38 00000000 |.........v......|
   0080 00000000 00000000 00000000 00000000 |................|
   0090 00000000 00000000 14098F09 22115700 |................|
   00A0 00000000 00000000 00000000 00000000 |................|
   00B0 00000000 00000000 00000000 00000000 |................|
   00C0 00000000 00000000                   |........        |

   RSCX     Resource Extension Block         Loc: 25A51B38
   ---- -------- -------- -------- --------  ----------------
   0000 C4C6E2D9 E2C3E740 00000060 00000000 |DFSRSCX ...-....|
   0010 CCFD1CA2 2AC53002 00000000 00000000 |...s.E..........|
   0020 00000000 00000000 00000000 00000000 |................|
   0030 40404040 40404040 00010001 40000000 |        .... ...|
   0040 00000000 00000000 00000000 00000000 |................|
   0050 00000000 00000000 00000000 00000000 |................|
   *14140/150239*
```

Figure 2-21 Output from a /DIAG command against a Full Function DB

Example showing Fast Path DB( ) control blocks

Figure 2-22 on page 50 shows the output from the following /DIAG command against a Fast Path DB:

```
/DIA SNAP DB(IVPDB3) SHOW(DDIR,RSCX,DMCB) OPTION(SYSOUT)
```

```
/DIAGNOSE SNAP STORAGE DISPLAY

   Resource: DB(IVPDB3)

   DDIR     Database Directory Block         Loc: 222868D8
   ---- -------- -------- -------- --------  ----------------
   0000 00000000 CD2D019A C9E5D7C4 C2F34040 |........IVPDB3  |
   0010 22252A60 000A0500 002C002C 0000000C |...-............|
   0020 00400000 00010200 03010000 00000000 |. ..............|
   0030 222869A0 00000000 00000000 C4000000 |............D...|
   0040 00200000 00000000 00000000 00000000 |................|
   0050 00000000 00000000 00000000 00000000 |................|
   0060 00000000 00000000 00000000 00000000 |................|
   0070 00000000 00000000 25A52008 00000000 |.........v......|
   0080 00000000 00000000 00000000 00008000 |................|
   0090 00000000 00000000 14098F09 22115700 |................|
   00A0 00000000 00000000 00000000 00000000 |................|
   00B0 00000000 00000000 00000000 00000000 |................|
   00C0 00000000 00000000                   |........        |

   RSCX     Resource Extension Block         Loc: 25A52008
   ---- -------- -------- -------- --------  ----------------
   0000 C4C6E2D9 E2C3E740 00000060 00000000 |DFSRSCX ...-....|
   0010 CCFD1CA2 2AC59678 00000000 00000000 |...s.Eo.........|
   0020 00000000 00000000 00000000 00000000 |................|
   0030 40404040 40404040 00010001 40000000 |        .... ...|
   0040 00000000 00000000 00000000 00000000 |................|
   0050 00000000 00000000 00000000 00000000 |................|

   DMCB     DEDB Master Control Block        Loc: 22252A60
   ---- -------- -------- -------- --------  ----------------
   0000 00000000 00000000 C9E5D7C4 C2F34040 |........IVPDB3  |
   0010 00020001 22253530 22253530 A2575000 |............s.&.|
   0020 C4C2C6C8 C4C3F4F0 24E63108 000002EE |DBFHDC40.W......|
   0030 22252AD4 222868D8 00000032 00011300 |...M...Q........|
   0040 00000000 00004000 00000000 00000000 |...... .........|
   0050 00000000 00000000 00000000 22252000 |................|
   0060 00000000 22253540 22252B04 00000000 |....... ........|
   0070 00000000                            |....            |
   *14140/151831*
```

Figure 2-22 Output from a /DIAG command against a Fast Path DB

2.5.9 /DIAGNOSE SNAP MSNAME( ) support added

Users can now gather diagnostic information for all control blocks that are associated with a specified MSNAME (logical link path) by using **/DIAG SNAP MSNAME**. Previously, no support existed. It now includes the following standard TM blocks and SHOW( ) option for filtering:

- SHOW(PRI)
- SHOW(blockname)
- SHOW(ALL)
- SHOW(OPT)

Benefits of support added

The value of this new parameter ensures consistency and simplifies data capture process. It requires less overall time and effort in analyzing and solving problems.

Potential users include IMS system programmers, support technicians, and developers.

Example 2-10 shows an example of this enhanced command.

Example 2-10 Example of /DIAGNOSE SNAP MSNAME( )enhanced command

```
/DIAGNOSE SNAP MSNAME(msname) SHOW(blockname)
```

2.5.10 /DIAGNOSE SNAP LINE( ) support for more blocks

Users can now gather diagnostic information for more control blocks that are associated with a communication line by using the **/DIAG SNAP** command that is issued for the LINE( ) resource.

The new SHOW( ) parameter values that were introduced in 2.5.4, “/DIAGNOSE SNAP ... SHOW( ) support added” on page 43 apply to the LINE( ) resource. Users can now filter the captured diagnostic data by the following parameters:

- SHOW(RECANY): Capture VTAM and receive any input and output buffer control blocks (RAQE and RAQERES blocks).
- SHOW(SA): A synonym for SHOW(SAVEAREA) to capture the save area control block (SAVEAREA block).

For more information about the control blocks that can be captured by using the **/DIAGNOSE SNAP LINE( )** command, see *IMS Version 13 Commands, Volume 1: IMS Commands A-M*, SC19-3648.

2.5.11 /DIAGNOSE SNAP LINK( ) support for more blocks

Users can now gather diagnostic information from more control blocks that are associated with a logical link by using the **/DIAG SNAP** command that is issued for the LINK( ) resource.

The new SHOW( ) parameter value applies to the LINK( ) resource. Users can now filter the captured diagnostic data by save area control block by using the SHOW(SA) filter.

SHOW(SA) is a synonym for SHOW(SAVEAREA) and captures the SAVEAREA block.

For more information about the control blocks that can be captured by using the **/DIAGNOSE SNAP LINK( )** command, see *IMS Version 13 Commands, Volume 1: IMS Commands A-M*, SC19-3648.

2.5.12 New messages added

Table 2-9 lists the new IMS 13 messages that were added with this **/DIAG** command enhancement.

Table 2-9 IMS 13 messages added with DIAG command enhancements

Messages	Description
DFS3613I	DGS TCB INITIALIZATION COMPLETE
DFS2857E	DIAGNOSE COMMAND INTERNAL ERROR - MOD=modname RSN=nnnn
DFS2858E	DIAGNOSE COMMAND SEVERE ERROR - reason text
DFS2859I	DIAGNOSE COMMAND UNSUCCESSFUL - reason text
DFS3785E	DIAGNOSE AWE INITIALIZATION FAILED - reason text
DFS3786E	DIAGNOSE AWE PROCESSING ERROR - reason text
DFS3787E	DIAGNOSE SYSOUT PROCESSING ERROR - reason text
DFS3788I	DIAGNOSE SYSOUT DATA SET dsname action FOR SNAP resource TKN(token)
DFS3789I	DIAGNOSE COMMAND SNAP resource QUEUED TO SYSOUT TKN(token)

2.5.13 DIAGNOSE command enhancements summary

The IMS **/DIAGNOSE** command is enhanced with new output options and to include information about the following resource types:

- SYSOUT option now available for **/DIAGNOSE SNAP** output
- **/DIAGNOSE SNAP** command extended to include more resources and more coverage of existing resources

  It also now supports over 300 blockname options across all resource types.

Benefits

This IMS 13 enhancement has the following benefits:

- Process for capturing diagnostic data that is used in troubleshooting IMS issues is simplified.
- Decreased time and effort is required to capture diagnostic information.
- Improved turn-around time in problem resolution.
- Cost effective, nondisruptive alternative to console dumps.

SYSOUT option

This option includes the following benefits:

- Documentation can be gathered and stored in a readable format that is easy to retrieve.
- Time-consuming SYSLOG searches and manual data formatting before transmission is no longer required.
- Fits with automated diagnostic data capture.
- Helpful for intermittent problems.
- Facilitates historical tracking of a resource.

/DIAG command expanded

The `/DIAG` command is expanded to include more resources and capture more blocks for existing supported resources. It also features the following benefits:

- Search overhead is reduced by allowing user to narrow search scope with RM( )
- More interactive and can be used more as a tool for easing the real-time diagnosis process

2.6 Concurrent application threads

IMS systems continue to grow, needing more dependent regions to process new workloads that are associated with usage of open database, internet connectivity options, and applications with longer dependent region residency times.

The number of dependent regions grew from 15 in IMS 360 to 4095 dependent regions in IMS 13.

IMS 13 quadrupled the maximum number of Partition Specification Tables (PSTs) that can run concurrently in a particular IMS image. It is used for the following applications:

- Active dependent regions (MSG/BMP/IFP/JMP/JBP)
- CICS/DBCTL threads
- Open Database Access threads

This enhancement includes the following related parameters:

- MAXPST= parameter
- PST= parameter
- MAXTHRDS= parameter
- MAXTHRDS= and MINTHRDS= parameters for DBCTL

2.6.1 MAXPST= parameter

The MAXPST= parameter is used in DBC, DCC, and IMS procedures. It specifies the maximum number of PSTs for an online IMS control region and includes the following parameters:

- Default is 255 (no change)
- Maximum value is 4095 in IMS 13 (quadrupled)

MAXPST= controls the maximum number of active dependent regions (MSG/BMP/IFP/JMP/JBP), CICS/DBCTL threads, and Open Database Access threads.

Important: Reducing MAXPST= requires a cold start.

2.6.2 PST= parameter

The PST= parameter is used in DBC, DCC, and IMS procedures, and specifies the number of PSTs that are allocated during IMS initialization. It includes the following parameters:

- Value can be between 0 and the MAXPST= value
- Default is 0

IMS adds PSTs up to the MAXPST= value for increasing system activity, and IMS deallocates PSTs and releases their storage as workload decreases to this PST= value.

2.6.3 MAXTHRDS= parameter

The MAXTHRDS= parameter is for Open Database, which is specified in the CSLDCxxx ODBM Configuration PROCLIB member. It specifies the maximum number of concurrent active threads for an individual datastore and includes the following parameters:

- Can be set globally (all datastore connections) or locally (specific datastore connection)
- Default value is 1
- Value can be between 1 and the MAXPST= value

2.6.4 MAXTHRDS= and MINTHRDS= parameters for DBCTL

The MAXTHRDS= and MINTHRDS= parameters for DBCTL are specified on the DFSPRP macro that defines the DRA start table.

MAXTHRDS= specifies the maximum number of DRA thread TCBs to be available at one time. MINTHRDS= specifies the minimum number of DRA thread TCBs to be available at one time. Maximum value is the MAXPST= value, and minimum value is 1.

2.6.5 ODBM MINTHRDS/MAXTHRDS enhancement

The default MINTHRD value of 1, which is used in an RRS=N environment, can result in an excessive number of attach and detach processes, which can result in performance degradation.

With this IMS 13 enhancement, the MINTHRD value in an RRS=N ODBM environment is set to approximately 62% of the value of MAXTHRD.

By establishing a higher minimum number of threads, the performance issues that are associated with attaching and detaching threads are reduced.

This enhancement was retrofitted in IMS 11 and IMS12 with the following APARs:

- IMS 11 APAR - PM63977 UK82609
- IMS 12 APAR - PM63976 UK82608

ODBM uses ODBA in the RRS=Y case. Because ODBA TCBs are used, thread TCBs are not attached or detached for the DRA.

MINTHRD default is still 1 for ODBA.

2.6.6 Commands type-2 and type-1considerations

The following DRD commands for transactions support a MAXRGN parameter of up to the MAXPST= value:

- CREATE TRAN
- CREATE TRANDESC
- UPDATE TRAN
- UPDATE TRANDESC

The `/CHANGE TRAN` command supports a MAXRGN parameter of up to MAXPST=.

The output of the `/DISPLAY TRAN` command supports a four-digit decimal number for BAL() instead of a three-digit number.

The TRANSACT IMSGEN macro supports MAXRGN=255 only.

2.6.7 Coexistence considerations

In a mixed-version IMSplex, commands that reference the MAXRGN parameter and specify a value greater than 999 pass validation at the Operations Manager (OM) layer because OM uses the highest IMS version grammar.

However, if any these commands are routed to an IMS system that is a lower version than IMS 13, the command fails on that IMS system. The following commands are affected:

- /CHANGE TRAN
- CREATE TRAN
- CREATE TRANDESC
- UPDATE TRAN
- UPDATE TRANDESC

2.6.8 Benefits of increasing the MAXPST= parameter

Because of the following benefits of concurrent thread enhancement by increasing the MAXPST= parameter, customers can now have increased capacity and scalability for their IMS systems:

- Larger capacity for mergers and acquisitions without having to add more IMS images.
- Increased workloads with latest zEnterprise® hardware with room for vertical growth.
- More regions for IMS 13 synchronous program switch function, also synchronous callout, distributed sync point, and so on, for longer region occupancies.

MAXPST should no longer be a limiting factor in IMS growth.

2.7 Reduced total cost of ownership

Reduced total cost of ownership (TCO) for IMS users is a major focus in IMS 13 because reducing the costs of running on the mainframe platform is an important continuing focus for all IBM products on this platform.

Cross-platform focuses on reducing mainframe software costs because they are based on CPU usage. Therefore, reducing CPU usage is major focus because of the potential performance benefits.

IMS laboratory reviewed the IMS code in the following areas:

- Reducing path length
- Optimizing frequently used processes
- Latch and lock improvements
- Storage reductions
- Use of zEnterprise hardware functions: New zEnterprise / z/OS capabilities

IMS 13 includes the following reduced TCO focus areas:

- Specific reduced TCO enhancements
- Several general internal enhancements

- IMS IBM System z Integrated Information Processor (zIIP) utilization enhancement
- External subsystem access facility (ESAF) for Java dependent regions

2.7.1 Specific reduced TCO enhancements

This section describes other important enhancements that are specific to TCO reduction.

IMS logger log latch contention reduction

Improves usage of log latch and log buffer management for increased logging bandwidth and more efficient processing.

Change the IMS logger so that log data can be moved to a buffer without requiring an exclusive latch to reduce contention on high activity systems.

Shared queues local first optimization

It now applies to program-to-program switch messages and ordinary input messages. It also avoids scheduling on another IMS when the local IMS can process the program-to-program switch message.

Transaction messages that are inserted by an application program now are considered for local first processing, subject to the same rules as transaction messages that originate from the network.

This configuration can improve performance by avoiding scheduling for a specific transaction instance on other IMS in the same shared queues group when the message can be processed by the local IMS.

Use of pageable 1 MB pages

It is based on usage of new zEC12 processors with Flash Express storage and z/OS 1.13, and it provides improvements in dynamic address translation and usage of translation lookaside buffer (TLB).

On zEC12 machines with Flash Express, z/OS supports 31-bit storage that is backed by 1 MB large real pages. This support was added to z/OS 1.13 and above.

Large page-backed storage improves performance by reducing the number of steps in dynamic address translation, and by improving translation lookaside buffer (TLB) coverage.

IMS 13 requests the CQS interface buffers to be backed by pageable large pages, when available.

IMS 13 also requests the following pools to be backed by pageable 1 MB pages. However, if you page fix these pools, they are not backed by 1 MB pages:

- DPSB pool (DLI/SAS PSB pool)
- DLDP pool (DMB pool)
- DBWP pool (DMB work pool)

DB space management block serialization latch improvements

IMS DB Block Serialization uses an internal chain of latches to ensure integrity when segments are inserted into or deleted from an IMS Full Function database. Normally, this latch is not a problem; however, it can be an issue when there is heavy BMP activity (multiple concurrently running BMP jobs that are performing insert and delete).

The chain of latches is protected by a latch header. Before IMS 13, there was only one latch header across all of IMS, which might result in significant contention and increased elapsed times during heavy insert and delete activity.

In IMS 13, the DB block serialization latch is split from a single latch header to multiple latch headers, which significantly reduces the likelihood of contention for this latch.

The reduction in contention can lead to shorter elapsed times and allow for more concurrency, especially when multiple concurrent BMP jobs are running with a large amount of insert and delete activity.

In addition to the performance aspects of this enhancement, there are new statistics that are externalized in the x'4507' log record that can be used to monitor the block serialization latch.

The IMS 13 DB Block Serialization Enhancement reduces internal latch contention and might result in lower elapsed times and reduced CPU cost for heavy batch (BMP) workloads. The Block Serialization enhancement yielded results of up to a 50% reduction in elapsed time with the concurrent run of BMP jobs, performing many inserts and deletes. The testing also showed the potential of CPU reduction by up to 11%.

MEMDSENQMGMT exploitation

The use of more efficient memory-based data set ENQ management improves allocation of large number of data sets with z/OS 1.12 and later.

IMS 13 issues an IEFDDSRV DSENQMGMT=MEMORY call from both control region and DLI/SAS to request memory-based data set ENQ management when z/OS 1.12 or later is used.

Memory-based management can be more efficient for jobs that allocate many data sets.

To enable this change be effective, add the following statement in your ALLOCxx SYS1.PARMLIB member:

```
MEMDSENQMGMT(ENABLE)
```

2.7.2 Several general internal enhancements

This section describes smaller IMS internal changes through several enhancements that are across many components of IMS, which shows the reduced TCO focus for the entire IMS product:

- Type-2 processing for the IBM IMS Universal Drivers
- OTMA YTIB chain changed from a single linked list to a hash table to improve FINDDEST performance
- Conversion of OTMA and IMS Connect STORAGE calls to CPOOL in its XCF SRB exits, which adds 167 MB of virtual storage in EVPT
- Remove unnecessary clearing of OTMA buffers
- Improved z/OS supervisor calls (SVCs) directory entry search algorithm in DFSCPY00 and removal of IVSK instructions
- CQS mainline modules changed to use branch-relative branching
- Cache efficiency improvements (DPST blocks packed into a single IPAGE to keep cache references localized)
- IMS page load service algorithm optimization

- IMS dispatcher optimizations
- General instruction optimization by replacing STCK with STCKF (long displacement facility usage)
- IMS cache manager spin loop elimination
- BPE dispatcher hot cache line prefetch
- Elimination of local lock obtain around buffer page fix for OSAM buffers

2.8 IMS zIIP utilization enhancements

> **Note:** The information in this section provides only general descriptions of the types and portions of workloads that are eligible to be run on Specialty Engines (SEs); for example, zIIPs, zAAPs, and IFLs. IBM authorizes customers to use IBM SE to run the only processing of Eligible Workloads of specific programs that are expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines”, which is available at this website:
>
> http://www.ibm.com/systems/support/machine_warranties/machine_code/aut.html
>
> No other workload processing is authorized for running on an SE. IBM offers SE at a lower price than General Processors or Central Processors because customers are authorized to use SEs to process only certain types or amounts of workloads as specified by IBM in the Authorized Use Table for IBM Machines.

Designated areas of IMS 13 processing can use IBM zIIPs as part of its overall focus on reducing TCO. A prerequisite to the running on zIIP is the enablement of tasks to be run as enclave service request blocks.

Certain processing in the IMS Connect address space and in the Open Database Manager (ODBM) address space can be run under enclave service request blocks (SRBs). IMS can direct z/OS to authorize such work to be processed on an available zIIP.

zIIP offload is off by default. You can turn it on conditionally (only active if one or more zIIPs are present at initialization), or unconditionally (zIIP-eligible threads run in SRB mode, even if there are no zIIPs, which allows for projecting amount of possible offload).

The purpose of this enhancement is to reduce the processing that is done on general processors to help reduce the overall total cost of computing for selected workloads, including IMS SOAP Gateway, direct IMS DB access, and MSC and ISC by using TCP/IP.

2.8.1 zIIP overview

From a functional perspective, a zIIP is a System z processor, similar to a general CP. It has all the same registers, instruction set, and so on.

The difference is how zIIPs get work, and how work is charged. zIIPs are processors that have work routed to them from general GPs. This routing is controlled by the workload manager, under direction from software. Consider the following points:

- There is no charge for software runs a zIIP.
- zIIP offloading is restricted to work that is running as an enclave SRB.

This means that normal TCB-mode programs cannot be dispatched on a zIIP.

The benefit to zIIP customers is that they are not charged for the portion of their workload that is offloaded to a zIIP. This can result in lower software license charges than if the work ran on a general CP.

Enclave overview

In the context that it is used here, an *enclave is* a workload manager term. It refers to a z/OS transaction that can span multiple dispatchable units (TCBs and SRBs).

Enclaves are a way to group a set of related activities in the processor together for reporting and management purposes.

Essentially, you create an enclave and then you schedule work into it. An enclave SRB, then, is a normal SRB that is scheduled into a previously created enclave.

z/OS uses enclaves as the connection between workload and zIIPs. Enclaves do not have to be associated with zIIP offloading, but they can be by completing the following steps:

1. Create an enclave by using the Workload Manager macro IWM4ECRE. The macro returns a token that identifies the enclave.
2. You can schedule work into the enclave and WLM manages and reports on it separately from any other work.
3. If you want to offload that work onto a zIIP, you issue the macro IWM4EOCT, which tells WLM that this enclave is eligible for zIIP offloading. From this point, any SRBs that are scheduled into this enclave are zIIP-eligible and can run on a zIIP processor.

Enclave SRB running

In IMS 13, certain processing in the IMS Connect address space and in the IMS Open Database Manager (ODBM) address space (as specified in this section) when enclave Service Request Block (SRB) running is enabled by a user is run under enclave SRBs.

Enclave SRB running can be enabled as unconditional or conditional based on the presence of at least one online zIIP during address space initialization.

In an unconditional status, the user can enable such processing to be run in enclave SRB mode when no zIIP is online or part of the user's System z environment, but no zIIP utilization can result.

A user might request enclave SRB running even in the absence of zIIPs to allow the system to project how much work is eligible to run on zIIP and probably can if there were zIIP capacity available.

For more information, see the PROJECTCPU parameter of the IEAOPTxx SYS1.PARMLIB member in the *MVS Initialization and Tuning Reference*, SA22-7592-23.

When enclave SRB running is enabled, IMS 13 directs z/OS to authorize such work to be processed on an available zIIP.

zIIP usage by IMS

The following IMS Connect and ODBM processing can run under an enclave SRB in IMS 13 and can use zIIP processors:

- The processing of IMS Connect and ODBM address space Distributed Relational Database Architecture (DRDA) threads for DRDA requests that are arriving via TCP/IP (DRDA workload).

- The processing of IMS Connect address space SOAP message threads for SOAP messages that are arriving via TCP/IP (SOAP workload) threads for MSC messages arriving via TCP/IP (MSC workload).
- The processing of IMS Connect address space Intersystem Communication (ISC) threads for ISC messages that are arriving via TCP/IP (ISC workload).
- The processing of ODBM address space threads for requests that are arriving through the CSLDMI API (CSLDMI workload).

Certain types of processing can run only in TCB mode. This includes the SRB-mode restricted items, such as z/OS SVCs, and code that is not owned by IBM, such as user exits.

User exits must always be called in TCB mode, so they cannot be run on a zIIP.

Also, certain processing cannot run under enclave SRBs because of technical restrictions. Such processing includes calling z/OS Resource Recovery Services (RRS), IMS DL/I call processing, and z/OS SVCs.

IMS switches from SRB mode into TCB mode to perform such processing; therefore, such processing does not run on a zIIP.

> **Note:** New fields are added to BPE statistics user exit records to report information about SRB-mode and zIIP-mode processing.

2.8.2 Environment and user control

You can use the BPE configuration parameter member of the IMS PROCLIB data set to define the BPE running environment settings, including the enablement of conditional enclave SRBs threads with the following option:

```
BPECFG= CONDSRB
```

The syntax of the CONDSRB (as shown in Figure 2-23 on page 61) makes available the following keywords:

- NEVER

  Never run such conditional SRB threads in SRB mode, even if there are zIIPs online. This is the default.

  Running on a zIIP in SRB mode can reduce software license charges. However, SRB-mode running is more restrictive than TCB mode.

  There can be some other processing overhead involved if a thread that is running in SRB mode must switch back to TCB mode to perform some processing that is not supported in SRB mode.

  Additionally, if you have IIPHONORPRIORITY=NO specified in the z/OS IEAOPTxx SYS1.PARMLIB member, ready zIIP-eligible work waits for a zIIP processor when one is not immediately available, even if there is a standard processor available that can process the work.

  For more information, see *MVS Initialization and Tuning Reference Manual*, SA22-7592.
- ALWAYS

  Always run such conditional SRB threads in SRB mode, even if there are no zIIPs online.

  This option is useful for projecting potential zIIP usage via the PROJECTCPU parameter of the IEAOPTxx SYS1.PARMLIB member.

For more information, see *MVS Initialization and Tuning Reference*, SA22-7592-23.

- COND

  Run such conditional SRB threads in SRB mode if there are zIIPs online.

  This option automatically activates the running of conditional SRBs when you start an IMS Connect or IMS ODBM address space on a System z LPAR with one or more online zIIP processors at the time BPE services were started, as shown in Figure 2-23.

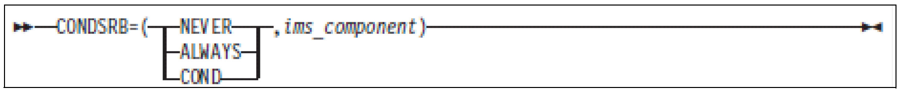


Figure 2-23 Options for CONDSRB

  If IMS detects no zIIPs, BPE runs the threads in TCB mode and no usage of a zIIP for such threads occurs.

  Any user exits that are called by zIIP-eligible threads do not run under an enclave SRB. User exits are always given control in TCB mode and such user exit instructions are not authorized to be processed on a zIIP.

The second parameter of CONDSRB specifies the following IMS component address space type to which the CONDSRB statement applies:

- HWS: IMS Connect
- ODBM: Open Database Manager

Create a BPE configuration proclib member or update an existing member for the IMS Connect or the IMS ODBM address space. Consider the following points:

- For IMS Connect, add the following statement to its BPE configuration proclib member:

  ```
  CONDSRB(COND,HWS)
  ```

- For IMS ODBM, add the following statement to its BPE configuration proclib member:

  ```
  CONDSRB(COND,ODBM)
  ```

- Start the IMS Connect or IMS ODBM address space with the BPECFG= parameter in the PARMS= parameter string of the address space start JCL. Specify the name of the created or updated BPE configuration proclib member.

2.8.3 z/OS maintenance considerations

z/OS APAR OA39392 (PTF UA66823) is recommended for z/OS 1.13 when you are running the IMS Connect or IMS ODBM address spaces in an environment where BPE conditional SRB threads are run in SRB mode, as shown in the following examples:

- When CONDSRB=COND is specified in the BPE configuration proclib member and one or more zIIPs is present
- When CONDSRB=ALWAYS is specified only in the BPE configuration proclib member

APAR OA39392 provides a new API for abnormally terminating SRBs. BPE SRB support internally uses this API during abnormal termination to attempt to ensure that all BPE SRBs end.

The APAR is needed only on z/OS 1.13. Later versions of z/OS have this support in the base code.

It is possible to run with BPE conditional SRB threads in SRB mode without the SRB abnormal termination API that is provided by OA39392. However, if a conditional SRB thread cannot end normally (for example, because it is in an infinite loop), the BPE address space might hang during termination while waiting for the SRB, which might force it exit from the system.

Forcing an address space can prevent cleanup of system resources, which might require a z/OS IPL to correct. For this reason, IBM recommends that you specify CONDSRB=NEVER for any IMS V13 IMS Connect or IMS ODBM address spaces that run on a z/OS without SRB abnormal termination support.

2.8.4 Benefits

Selected workloads, including IMS SOAP Gateway, direct IMS DB access, and MSC and ISC by using TCP/IP can use zIIP processors, which can lower software costs.

For more information about performance considerations, see 9.6, “zIIP performance considerations” on page 247.

For more information about IBM System z Integrated Information Processor (zIIP), see the following System z website:

http://www.ibm.com/systems/z/ziip/

2.9 ESAF support in Java dependent regions

This section describes new support for using the External Subsystem Access Facility (ESAF) in Java dependent regions.

Before IMS 13, the only external subsystem (ESS) that Java dependent regions (JDR) applications accessed was DB2 by using the DB2 RRS attach facility (RRSAF).

There was no access to other external subsystems, such as WebSphere MQ. DB2 RRSAF usage was unique to JDR versus other region types. No access to other external subsystems required more complex external subsystem definitions.

There was a need for consistent ESAF interface across all region types for DB2 and for less complex external subsystem definitions.

2.9.1 JDR ESAF enhancement in IMS 13

With IMS 13, the following methods are available for accessing DB2 from JDRs:

- Via the existing DB2 Resource Recovery Services Attach Facility (RRSAF) interface
- Via the standard ESAF interface

There is support for the SSM= parameter on the JMP/JBP dependent region start JCL, but it is only allowed one ESS connection method per JMP/JBP.

The default ESS connection method is DB2 RRSAF and there is no effect on existing users. Also, ESAF must be set as the connection method by specifying SSM= in the JMP/JBP dependent region JCL.

This is another method for accessing DB2 and the same method as accessing DB2 from non-JDR region types. It provides a more efficient and less complex connection method for DB2 access.

IMS 13 ESAF interface

With IMS 13, the ESAF interface can be used in JMP/JBP regions to access any ESAF defined to the IMS control region, such as WebSphere MQ, DB2, and WebSphere Optimized Local Adapter (WOLA).

There is support for the SSM= parameter on the JMP/JBP dependent region start JCL, and there is only one ESS connection method that is allowed per JMP/JBP.

The default ESS connection method is DB2 RRSAF with no effect on existing users. ESAF must be set as the connection method by specifying SSM= in the JMP/JBP dependent region JCL

Support for all types of ESAF interfaces is provided. WebSphere MQ and WOLA can now be accessed via JMP/JBP regions.

Enabling ESAF for JMP or JBP regions

To enable the ESAF for a JMP or JBP region, the SSM= parameter must be specified on the DFSJMP procedure or DFSJBP procedure. Consider the following points:

- The SSM= value can be the same value as specified on the SSM= parameter for the IMS procedure.
- Ensure that the DFSESL DD statement is included in the DFSJMP/DFSBJP procedure or have these libraries available via JOBLIB/STEPLIB.

The ESAF set up process is available in the IMS library and the Information Center.

2.9.2 DFSJMP procedure that uses ESAF

Example 2-11 shows the DFSJMP procedure for the use of ESAF.

Example 2-11 DFSJMP procedure

```
//         PROC SOUT=A,RGN=512K,SYS2=,
//              CL1=001,CL2=000,CL3=000,CL4=000,
//              OPT=N,OVLA=0,SPIE=0,VALCK=0,TLIM=00,
//              PCB=000,STIMER=,SOD=,
//              NBA=,OBA=,IMSID=,AGN=,
//              PREINIT=,ALTID=,PWFI=N,APARM=,
//              LOCKMAX=,ENVIRON=,
//              JVMOPMAS=,PRLD=,SSM=,PARDLI=
//*
//JMPRGN EXEC PGM=DFSRRC00,REGION=&RGN,
//              TIME=1440,DPRTY=(12,0),
//              PARM=(JMP,&CL1&CL2&CL3&CL4,
//              &OPT&OVLA&SPIE&VALCK&TLIM&PCB,
//              &STIMER,&SOD,&NBA,
//              &OBA,&IMSID,&AGN,&PREINIT,
//              &ALTID,&PWFI,'&APARM',&LOCKMAX,
//              &ENVIRON,,&JVMOPMAS,&PRLD,
//              &SSM,&PARDLI)
//*
```

```
//*
//STEPLIB  DD DSN=IMS.&SYS2.PGMLIB,DISP=SHR
//         DD DSN=IMS.&SYS2.SDFSJLIB,DISP=SHR
//         DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//         DD DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD DSN=IMS.&SYS2.PROCLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=&SOUT,
//         DCB=(LRECL=121,BLKSIZE=3129,RECFM=VBA),
//         SPACE=(125,(2500,100),RLSE,,ROUND)
```

Example 2-12 is a DFSESL DD statement that can be added if these libraries were not available via JOBLIB/STEPLIB.

The external subsystem libraries must follow IMS.SDFSRESL.

Example 2-12 DFSESL DD statement

```
//DFSESL   DD DISP=SHR,DSN=IMS.SDFSRESL
//         DD DISP=SHR,DSN=DSNxxx.DSNLOAD
//         DD DISP=SHR,DSN=DSNyyy.DSNLOAD
```

2.9.3 DFSJBP procedure that uses ESAF

Example 2-13 is the DFSJBP procedure for the use of ESAF.

Example 2-13 DFSJBP procedure

```
//       PROC MBR=TEMPNAME,PSB=,JVMOPMAS=,OUT=,
//            OPT=N,SPIE=0,TEST=0,DIRCA=000,
//            STIMER=,CKPTID=,PARDLI=,
//            CPUTIME=,NBA=,OBA=,IMSID=,AGN=,
//            PREINIT=,RGN=512K,SOUT=A,
//            SYS2=,ALTID=,APARM=,ENVIRON=,
//            LOCKMAX=,PRLD=,SSM=
//*
//*
//JBPRGN EXEC PGM=DFSRRC00,REGION=&RGN,
//            PARM=(JBP,&MBR,&PSB,&JVMOPMAS,&OUT,
//            &OPT&SPIE&TEST&DIRCA,
//            &STIMER,&CKPTID,&PARDLI,&CPUTIME,
//            &NBA,&OBA,&IMSID,&AGN,
//            &PREINIT,&ALTID,
//            '&APARM',&ENVIRON,
//            &LOCKMAX,&PRLD,&SSM)
//*
//STEPLIB  DD DSN=IMS.&SYS2.SDFSJLIB,DISP=SHR
//         DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
//         DD DSN=IMS.&SYS2.PGMLIB,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//         DD DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD DSN=IMS.&SYS2.PROCLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=&SOUT,
//         DCB=(LRECL=121,RECFM=VBA,BLKSIZE=3129),
//         SPACE=(125,(2500,100),RLSE,,ROUND)
```

Similar to the DFSJMP procedure, you add a DFSESL DD statement if these libraries were not available via JOBLIB/STEPLIB. Also, the external subsystem libraries must follow IMS.SDFSRESL.

2.9.4 Benefits of ESAF support in Java dependent regions

JMP/JBP regions can now use the standard ESAF for accessing external subsystems, such as DB2 for z/OS, WebSphere MQ, and WOLA.

This configuration provides a consistent ESAF interface for DB2 across all region types and uses simplified external subsystem definitions. It also provides easier implementation than DB2 RRS Attach Facility (RRSAF).

This is a more efficient interface compared to the use of the DB2 RRSAF for DB2 access. It provides Java Message Services (JMS) access to WebSphere MQ from IMS Java applications. It also provides ESAF access to WebSphere MQ or WOLA from a COBOL or PL/I program that was called from the JMP/JBP Java application that is running in the Java Dependent Region.

2.10 IMS 13 service process changes

This section describes the latest IMS 13 enhancements that are included by services.

2.10.1 IMS Timing Services and connecting to External Subsystems

IMS timing is established in BMP regions by using CPUTIME= and the STIMER= parameters on the BMP start.

MPP regions establish timing by using the STIMER= parameter on the MPP start and the PROCLIM=parameter of the TRANSACT macro for the transaction that is processed.

Current

Before this change, application programs that were running in IMS dependent regions by using STIMER= were not stopped with ABENDU240 while in a long-running call to an External Subsystem. Instead, ABENDU240 was delayed until after the External Subsystem returned to IMS.

After apply APAR PM86872

With this maintenance, ABENDU240 is enforced in IMS dependent regions that are running in an External Subsystem (ESS) when time expires by using IMS Timing Services.

Now, when the timeout routines detect processing in an ESS, they schedule ABENDU240.

> **Note:** IMS establishes a time limit that uses the STIMER macro and the TASK parameter to indicate that the time interval is only decreased when the associated task is running.
>
> Any ESS that WAITs or SUSPENDs in the IMS dependent region does not decrease the time interval during these events and does not drive STIMER exit routines to issue abends.

2.10.2 Persistent JVM in MPP/BMP/IFP with DB2 calls through DB2 JCC

These services apply for users of persistent JVM in MPP/BMP/IFP where Java application makes DB2 calls through DB2 JDBC driver (JCC).

Current

For an IMS MPP/BMP/IFP application that is running in a Persistent JVM environment and calling a Java application that makes DB2 calls through DB2 JCC, there exists a potential performance impact because of the DB2 JCC connection not being reused if it can be reused.

The DB2 JCC connection can be reused if the transaction origin user ID did not change between transaction boundaries.

IMS controls the DB2 JCC connection during IMS sync point processing by starting the DB2 JCC method ExternalOps.reset() to tell DB2 JCC to reset the connection in preparation for the next transaction.

After apply of APAR PM81408

With this maintenance, IMS introduces user option DB2JCC_CONN_REUSE=Y/N, which is specified in the ENVIRON= proclib member parameter description of the dependent region (for instance, member DFSJVMEV).

IMS uses a DB2 JCC function with which the DB2 JCC connection can be reused if the transaction user ID was not changed between transactions.

The new DB2JCC_CONN_REUSE= Y/N parameter specifies whether (Y) or not (N) you want IMS to communicate to DB2 JCC that the DB2 JCC connection should be reused if possible for the next transaction. The default is N.

This option is recognized only for MPP, BMP, and IFP

If you specify DB2JCC_CONN_REUSE=Y, the following maintenance is required:

- DB2 APAR PM77184 for JCC3.64
- DB2 APAR PM77185 for JCC4.14

Note: If DB2JCC_CONN_REUSE=Y is specified, the DB2 JCC connection is reused only if the transaction user ID for the next transaction is the same as the user ID for the previous transaction.

Updated IMS Messages and Codes manuals

The information in this section was added in the IMS 12 and IMS 13 Messages and Codes manuals.

Messages and Codes, Volume 3: IMS Abend Codes

Under the IMS Abend U0101 description, under section *For DFSPCC20*:

```
Reg7 and Reg8 together contain the value of field RCPGM(program name) at time
of abend
```

Reason code 5 (Reg15=X'5'):

```
If Reg3=X'OC', message DFS650E precedes this abend
```

Messages and Codes, Volume 1: DFS Messages

DFS650E NON-LE COMPLIANT PROGRAM IN PERSISTENT JVM ENVIRONMENT, NAME=program name

- Explanation

  The user attempted to load and run a non Language Environment compliant or conforming program in a persistent JVM-dependent region environment. Any program that runs in a persistent JVM-dependent region environment must be Language Environment compliant or conforming.

- System action

  ABENDU0101-05 (reason code 5) is issued to stop the application.

- Programmer response

  Correct the program to make it Language Environment compliant or conforming. The program did not contain a valid Language Environment entry prolog. Make sure that the program was compiled with a current Language Environment enabled compiler. Foe more information about Language Environment compliant or conforming programs, see *z/OS Language Environment Programming Guide,* SA22-7561.

Performance enhancement for DB2 JCC

Code was modified to use a DB2 JCC function that allows the DB2 JCC connection to be reused when the user ID changes across individual transaction messages.

APAR PI15675 (open at the time of this writing) adds this functionality. When the transaction user ID changes, IMS no longer issues a DB2 JCC FULL reset (that is, ExternalOps.reset()),. Instead, it issues the new DB2JCC_CONN_REUSE option, which effectively allows the DB2 JCC connection to be reused when the transaction user ID changes across a transaction boundary.

2.10.3 Persistent JVM in MPP/IFP/BMP regions specifying CANCEL_PGM=Y

This maintenance applies for users of persistent JVM feature in MPP/BMP/IFP regions with the CANCEL_PGM parameter in the proclib member that is specified by the ENVIRON parameter. Consider the following points:

- ENVIRON= is a required parameter that specifies the name of the PROCLIB member that contains the environment settings.
- Environment variables can be specified are in the form of X=Y, where X is the environment variable and Y is the value of the environment variable.

Current

Currently, CANCEL_PGM=Y works only on an application schedule boundary (that is, unit of work [UOW] boundary) which means that CANCEL_PGM=Y takes effect only after the application ends.

CANCEL_PGM=Y must also work on a message/transaction/commit boundary (that is, UOR boundary), which means that CANCEL_PGM=Y must take effect on a sync point commit boundary while the application is running.

For example, within the applications message GU loop for a message- driven application, CANCEL_PGM=Y is honored.

As another example, for a non-message driven BMP, CANCEL_PGM=Y is honored per sync point commit cycle (that is, per SYNC call) while the application is running.

This model features the following differences:

- On the UOR boundary, CANCEL_PGM=Y does not act upon the running *main* application in the dependent region, which is the actual application that IMS schedules.
- On the UOW boundary, CANCEL_PGM=Y does act upon the main application.

After apply APAR PM90903

With this maintenance, you can specify whether you want IMS to "clean up" your COBOL programs and subprograms per commit cycle and across application program schedules. Consider the following points:

- CANCEL_PGM=Y works on a message/transaction/commit UOR boundary and a UOW boundary
- CANCEL_PGM=N is the default value

> **Note:** IMS does not "clean up" the program and all its subprograms. Working storage areas remain intact for the next program run.

2.10.4 PARDLI capability in the MPP, JMP, and IFP regions

PARDLI=0 / 1 specifies the parallel option to the running of the DL/I processing, where:

- 0 is the DL/I processing to be performed within the region. This is the default.
- 1 is the DL/I processing for this region that is to be performed in the IMS control region.
- If data capture (EXIT= on the DBD statement) is enabled and this is a delete, replace, or insert call, PARDLI=1 is ignored for the DL/I call.

Current

For BMPs, PARDLI=1 means all DL/I processing is to be performed in the IMS control region to prevent control region system 113 abends that result from system X22 abends in the BMP region. However, PARDLI=1 is not supported for other dependent region types.

After apply APAR PM78158

This maintenance specifies the PARDLI parameter for JMP, MPP, and IFP regions.

Use of PARDLI=1 for MPP, JMP, or IFP regions is intended only for application debugging purposes, if needed.

> **Important:** Using PARDLI=1 for MPP, JMP, or IFP regions can seriously degrade performance.

3

Transaction Manager enhancements

This chapter describes the changes in IMS Version 13 that are specific to the IMS Transaction Manager component. These changes include various Open Transaction Manager Access (OTMA) enhancements, the non-discardable Message Exit (DFSNDMX0) enhancements, and enhancements to Multiple Systems Coupling (MSC) and Intersystem Communication (ISC) connectivity. The main line item for ISC is a new option that supports TCP/IP network connectivity for ISC connections.

In the effort of reducing the total cost of ownership (TCO), there is an enhancement for the shared queues users, which can improve the performance. Shared queues local first optimization now applies to program-to-program switch messages and ordinary input messages. This configuration can improve performance by avoiding scheduling for a specific transaction instance on other IMSs in the same shared queues group when the message can be processed by the local IMS. Because the changes related to this enhancement are internal to IMS, we do not provide details in this book.

The chapter includes the following topics:

- Open Transaction Manager Access enhancements
- Non-discardable Message Exit (DFSNDMX0) enhancements
- Multiple Systems Coupling link increased maximums
- Intersystem Communication over TCP/IP

3.1 Open Transaction Manager Access enhancements

IMS 13 contains the following enhancements to Open Transaction Manager Access (OTMA):

- Early termination notification

  Autonomic enhancement for higher availability that through which OTMA clients can be informed of an IMS shutdown to choose more timely alternatives. This enhancement can reduce unsuccessful attempts to send in new requests and expedite shutdown processing.
- OTMA can leave XCF group earlier in the termination process.
- Global Flood Control can enforce global flood limit, including the ability to address the global flood condition rather than be warned only.
- Display HOLDq versus PRIMARY queue OUTPUT counts. This enhancement provides a mechanism to differentiate between CM0 versus CM1 output message counts for HOLDq-capable clients.
- Enhances automation with more console write to operator (WTO) messages.
- Addresses previous timing issues between OTMA C/I and client for asynchronous messages.
- OTMA client type is available.
- Enhancements to OTMA destination descriptors for asynchronous callout. Simplifies asynchronous messaging to WebSphere MQ and override request for Exit Routines.
- Small programming improvements.

These enhancements are described next.

3.1.1 Early termination notification

IMS 13 introduces an enhancement with which OTMA clients (for example, IMS Connect, WebSphere MQ, and OTMA C/I) can be notified of IMS termination earlier in the process for scheduled and unscheduled terminations. The IMS OTMA clients can then route new requests to other IMS systems to be processed. This enhancement addresses environments that process high rates of transactions via IMS OTMA and experience some transactions being accepted but not processed by IMS during scheduled or unscheduled IMS terminations.

This enhancement is autonomic for higher availability. *Autonomic* means here that there are no definition or configuration changes to use this enhancement. By using this enhancement, OTMA clients can be informed of an IMS shutdown to choose more timely alternatives. This ability potentially reduces unsuccessful attempts to send in new requests, and thus can expedite the shutdown process.

This enhancement also is retrofitted to IMS 12 by the APAR PM73869.

3.1.2 Enhancement to allow OTMA to leave XCF group earlier during the termination process

OTMA Early Termination Notification is part of the base IMS processing and is another autonomic enhancement.

When IMS features a planned shutdown (`/CHE {FREEZE | DUMPQ | PURGE}`), OTMA adds notification to OTMA clients during Phase 1 of the shutdown. OTMA sends a TMAMMNTR (x'3C') resource monitor protocol message with TMAMRSIM_S1SHTDN indicating IMS shutdown with TMAMRSIM_STATUS set to '1', which indicates IMS cannot process new requests.

OTMA also sends a TMAMCSPA (x'14') protocol message that informs clients to suspend processing for all TPIPEs. These protocol messages (which are sent at the start of shutdown) inform OTMA clients that IMS is no longer available and they must route new requests to another system.

Note: The resource monitoring protocol was introduced in IMS 11 as a way for OTMA to inform its member clients of potential issues.

When IMS has an unplanned shutdown (abend), OTMA issues IXCLEAVE to leave the XCF group immediately after the log buffers are purged. This action informs the OTMA clients via their XCF group exits that IMS OTMA is no longer active in the XCF group. The OTMA clients can then route new requests to another system.

3.1.3 Enhancement to Global Flood Control to enforce global flood limit

IMS OTMA also provides a new option to enforce the global flood limit to reject new requests and protect IMS from storage exhaustion. Previous releases protected flood conditions for individual OTMA clients (TMEMBERs), but provided warning messages only when the global flood limit was reached. With this enhancement, a user can address the global flood condition rather than receive only a warning.

In previous releases, OTMA's Message Flood Detection and Control capability provided a mechanism to automatically monitor the growth of active OTMA input messages and the control blocks that are associated with these input requests. Specifically, when an OTMA member or client sends a transaction to IMS, OTMA internally creates a control block that is called the Transaction Instance Block (TIB) to track each active input message. If several thousand OTMA input transactions are received from an OTMA member and are waiting to be processed, thousands of control blocks that represent the requests take much IMS storage, which can affect the overall IMS operations in the system. To prevent this type of OTMA message flood condition from an OTMA member, OTMA stops receiving the input transactions from this member that is based on a maximum value for the number of TIBs allowed for the OTMA member in the system. However, suppressing the input OTMA transactions targets only individual OTMA members that have the maximum value or threshold of TIBs that are defined.

This type of OTMA flood control does not protect the IMS system when it has multiple OTMA members. In this case, each of the members might not reach the flood limit for the member. However, the total TIBs from all the OTMA members can exceed the global flood limit of the IMS system. When this condition occurs, the IMS system (pre-IMS13) issues a warning signal with the Global Flood Control capability that was introduced in IMS 11 to the system console, the MTO, and all of the OTMA members without suppressing the input OTMA transactions. However, new transactions, can still come into IMS and flood the system, which can cause the IMS system to fail with a S878 abend.

The default OTMA global flood limit of the system was changed from 8,000 to 10,000 along with a default relief level change of 6400 - 5000. A new DFS3428W warning message is introduced when the 80% of global flood limit is reached. This message is issued at every 5% increment thereafter until the global flood limit is reached.

After the global flood limit is reached, the enhanced warning message DFS4388W is sent to the system console and MTO along with OTMA protocol messages that reflect a warning status to all the OTMA members when the global flood control is not yet activated. This global flood status can be relieved when the input messages in the system are processed and the number of TIBS is reduced to 50% or less of the global limit. The enhanced relief message DFS0793I is then sent to the IMS MTO and system console along with OTMA protocol messages, which reflects a good status to all the OTMA member clients.

These default actions occur if IMS is activated without limits to request input rejection with global flood control. This means that an IMS system without any specification or changes comes with the default OTMA global flood monitoring and a default global limit of 10,000. IMS sends out warning messages DFS3428W and DFS4388W to the system console and MTO along with protocol warning messages to all the OTMA members. No input OTMA transactions are suppressed. This default mode of OTMA global flood monitoring can be changed to suppress input transactions from all the OTMA members after a global flood control and limits are specified.

IMS 13 allows the activation of global flood control by using the following methods:

- Issuing the `/STA TMEMBER ALL INPUT ####` command where `####` is the global flood limit as in previous releases but now causes a rejection of new input when the value is reached in IMS 13.
- Specifying a global limit value in a new OTMA client descriptor (DFSOTMA).

The valid global flood limit has a new maximum value and can be specified as 0 - 99999. (The valid range for the member flood limit is 0 - 9999.) As in previous releases, a nonzero value activates the global flood control for OTMA for the system. A value of 0 deactivates the global flood support and a value of 1 - 200 is treated as though 200 was specified.

In IMS 13, the meaning and actions that are associated with the **`/STA TMEMBER ALL INPUT ####`** command changed. In IMS 11, when this command was issued, OTMA used the specified limit instead of the system default of 8000 to monitor the total active input message count for send-then-commit (CM1) transactions from all of the OTMA members. If the total number reached this specified limit, OTMA issued a DFS4388W to the IMS MTO and system console along with OTMA protocol messages with the warning status to all of the OTMA clients. In IMS 13, the limit is used for more than a warning and allows OTMA to prevent any new message from being accepted until the flood situation is relieved.

The **`/DISPLAY OTMA`** command of a degraded system shows “SERVER+FLOOD” in the user status of the OTMA server member when the global flood control is activated and the global flood limit is reached. The **`/DISPLAY OTMA`** command is also enhanced so that the length of TIB and INPUT columns in output for this command increased by one. These fields can now display up to 5 digits of OTMA message numbers.

The option also exists to use a descriptor to define the global flood control limit by using a client descriptor. A special system client name, DFSOTMA, was introduced for this purpose. Activation of this DFSOTMA client descriptor enables the global flood control and requests suppression of any new OTMA transactions when the limit is reached. The parameter that applies to this function in the DFSOTMA client descriptor is INPT=. The value for this parameter can be up to 9999 (4 digits) for individual members and up to 99999 (5 digits) for DFSOTMA global setting. If the value is 1 - 9999 or 1 - 99999, it indicates the maximum number of input messages from all the OTMA members that can be waiting at the same time to be processed. If you specify a value of 0, OTMA deactivates the global flood monitoring and control. If the value is 1 - 200, it is treated as 200. Specification of other parameters, such as DRU= and T/O=, are ignored. The MAXTP parameter capability is described in “OTMA Transaction Expiration and Shared Queues SPE” on page 83.

Example 3-1 shows how the global limit is defined in DFSYDTx PROCLIB member.

Example 3-1 Example of defining the global flood control limit in DFSYDTx descriptor member

```
M DFSOTMA  INPT=22222
```

Global flood control has action points at 80% (warning), 100% (rejection of new messages), and 50% (relief) of the global flood limit. When global flood control is activated and the global flood limit is reached, OTMA rejects all of the new input transactions from all the members with OTMA sense code x'0030' and reason code x'0002'. Any synchronous program switch requests from the internal OTMA member DFSYICAL also are rejected.

The new DFS3429E error message is then sent to the system console and MTO and a protocol message with the command type set to X'3C' is sent to all the OTMA members with an "unavailable for work" status, unlike previous releases that sent only a "warning" status. When OTMA members receive this status, they can choose to take corrective action.

One example is the rerouting of all the new transaction requests from one IMS system to another. After IMS reaches the 50% value of the global limit for unprocessed messages, the global flood status is considered to be relieved and IMS begins accepting new input messages. The enhanced relief message DFS0793I is then sent to the IMS MTO and system console along with OTMA protocol messages, which reflects a good status to all the OTMA member clients.

Note: The MEMBER flood limit (versus GLOBAL for all members combined) remains at 9999, as in previous releases. When a member reaches the member flood limit, new transactions from this member are rejected with OTMA sense code x'0030' and the new reason code x'0001'.

Older releases of IMS provided the following messages for the input message flood condition:

- DFS1988W
- DFS1989E
- DFS0767I

IMS 11 introduced the following new messages:

- DFS4380W
- DFS4381I
- DFS4388W
- DFS0793I

In IMS 13, the existing DFS4388W (when limit is reached and the new global support is not active) and DFS0793I (when flood is relieved) messages are enhanced to make them clearer and consistent with the following new messages:

```
DFS3428W THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED XX% OF THE GLOBAL LIMIT
ZZZZ
DFS3429E THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED THE GLOBAL LIMIT ZZZZ
```

Before IMS 13, running the **/STA TMEMBER ALL INPUT** command enabled global flood notification. The same command in IMS 13 enforces the global limit and causes any new input to be rejected.

If DFSOTMA is used, IMS considers it to be one entry toward the 255 maximum member entries, which means that the total entries that are allowed for the OTMA clients or members become 254.

Figure 3-1 shows the action points of the global flood support before IMS 13 and the percentages at which WTO and X’3C’ protocol action messages are issued. The threshold values that are shown are the defaults of 5000 for an individual member and 8000 for all members globally.

OTMA Global Flood Control – Migration (Cont’d)

- ACTION points (pre-IMS 13)

Member	Default Threshold	80%		(at 5% incr) 85%-95%		100%		Relief	
		WTO MTO	3C	WTO MTO	3C	WTO MTO	3C	WTO MTO	3C
Individual Member Level	**5000**	X	X	X		X	X	at 50% X	X
Global Level - All Members	**8000**					X	X	at 80% X	X

- Member level
  - At 100% of the member threshold, messages are rejected until level is relieved at 50%
  - /DIS TMEMBER shows a FLOOD condition at 100%
- Global level
  - At 100% of the global threshold, a warning is sent but messages continue to be accepted

Figure 3-1 Action points of the global flood support before IMS 13

At the Member level, the WTO message and the X’3C’ message are sent at 80% as a warning and at 100% to indicate a severe condition. At increments of 5% from 85% - 95%, only the WTO message is sent. At 50% after the threshold was reached, the WTO and X’3C’ messages are issued to indicate a relief of the situation.

At the Global level, the WTO and X’3C’ messages are sent at the 100% mark and at 80% to indicate a relief of the situation.

In IMS 13, the default for global flood controls continues to be a warning message, but the default threshold is now at 10000 and the relief at 50%. However, if the global flood limits are activated by specification of the DFSOTMA client member descriptor or the **`/STA TMEMBER ALL INPUT`** command, reaching the 100% mark results in rejection of new transaction messages until the flood is relieved at 50%. Also, warning messages are issued at the 80% level, as shown in Figure 3-2 on page 75.

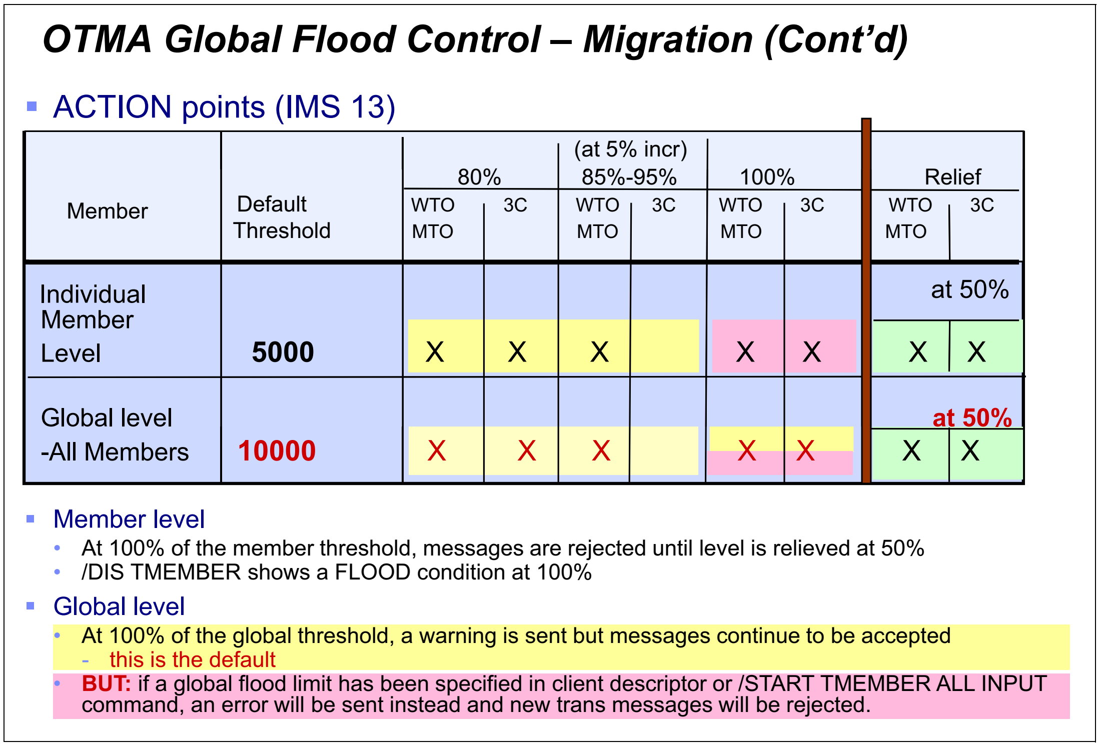


Figure 3-2 Action points of the global flood support before IMS 13

3.1.4 Ability to display HOLDq versus PRIMARY queue OUTPUT counts

IMS OTMA provides two queues for asynchronous output destinations that support "HOLDq"-capable clients (for example, IMS Connect). Send/Receive responses for Commit Mode 1 (Commit-then-Send) transactions are queued to the PRIMARY queue. Asynchronous messages (from SendOnly transactions or Alternative IOPCB inserts) are queued to the HOLDq queue and are retrieved by using Resume TPIPE protocol. HOLDq is a TPIPE secondary queue that is used for the following types of messages:

- ALTPCB messages that are chained to the first control block that is used for IOPCB output
- Send-only messages
- Rejected IOPCB output

Before IMS 13, the existing **`/DISPLAY`** command output reports the total messages that are queued to both queues. There is no external method to determine whether the messages are on the PRIMARY, HOLDq, or both.

IMS 13 adds a new OUTPUT parameter to the **`/DISPLAY TMEMBER TPIPE`** command to report the PRIMARY and HOLDq counts separately. This configuration can be helpful when potential bottlenecks are analyzed. The enhancement is valid only for HOLDq-capable OTMA clients and does not apply to non-HOLDq clients, such as WebSphere MQ. This option also is only available for non-shared queues IMS systems and is ignored when IMS has shared queues that are enabled. The OUTPUT parameter is mutually exclusive with the SYNC and QCNT parameters.

Figure 3-3 on page 76 shows an example of the new syntax of the **`/DIS TMEMBER TPIPE`** command with the new OUTPUT parameter.

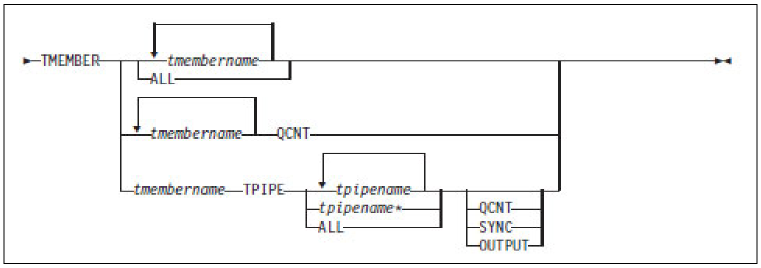


Figure 3-3 Syntax of the /DIS TMEMBER TPIPE command

Figure 3-4 shows an example of the OUTPUT display. For HOLDq client output, the ability to report on the PRIMARY and HOLDq queues might result in an increase of the display output. Specifying the OUTPUT parameter doubles the number of lines for the display. There might be some impact when the ALL or masked TPIPE names for TMEMBERs are used with many TPIPEs.

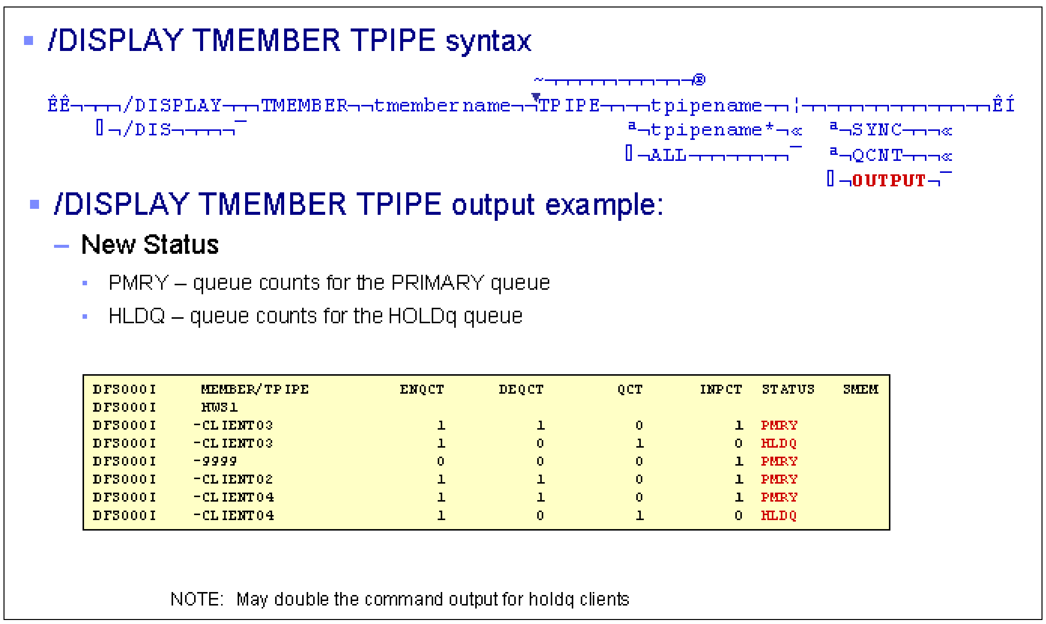


Figure 3-4 Example of the OUTPUT for /DIS TMEMBER TPIPE command

3.1.5 WTO messages for automation

There are many important IMS OTMA messages that are sent only to the Master Terminal Operator (MTO) LTERM. Many environments that use automation for console Write To Operator (WTO) messages and can also benefit from these OTMA messages. They are now sent to the MTO and WTO to the console. There are no changes to the message content. The following messages are used:

```
DFS0062W LOAD FAILED FOR DRU exit
DFS1281E (N)ACK ON NON-EXISTING TPIPE= tpipe FROM MEMBER= member
DFS1283E (N)ACK ON NON-EXISTING BLOCK IN TPIPE= tpipe MEMBER= member TOKEN= token
DFS1284E (N)ACK ON NON-WAITING BLOCK IN TPIPE= tpipe MEMBER= member TOKEN= token
DFS1297E OTMA TRANCOMPLETED. NO OUTPUT DUE TO MSG SIZE EXCEEDED 32K (32767)
DFS1964E DESTINATION TMEMBER ... TPIPE or DESTINATION NAME= nn NOT ACCESSIBLE,
DFS2374W OTMA GROUP NAME HAS CHANGED FROM oldname TO newname
DFS2375W IMS HAS CHANGED ITS OTMA MEMBER NAME FROM oldname TO newname
```

```
DFS2376W IMS OTMA MEMBER NAME name IS THE SAME AS A PREVIOUSLY ACTIVE CLIENT
DFS2386I OTMA IS CONNECTING TO THE MEMBER xxxxxxxx
DFS2391I MESSAGE DEQUEUE FAILED DURING RESYNCH FOR TMEMBER =x TPIPE = x
DFS2393I SRVRESYNCH REJECTED BY TMEMBER= xxxxxxxxxxxxxxxx
DFS2396I RESET SEQUENCE NUMBER FAILED DURING RESYNCH FOR TMEMBER = TPIPE =
```

The audit log function for the Operations Manager (OM) did not change and continues to be controlled by the UOM parameter in the DFSDFxxx member of PROCLIB for unsolicited messages.

3.1.6 OTMA Callable Interface (OTMA C/I) asynchronous enhancement

OTMA Callable Interface (OTMA C/I) provides a high-level interface for host infrastructure applications to send and retrieve messages with IMS OTMA by using XCF services. Before this enhancement, OTMA C/I asynchronous (CM0 – Commit-then-Send) messages that are sent by using otma_send_async_API have the responses sent immediately by OTMA because there is no HOLDQ. OTMA C/I must store these messages in XCF until the client issues otma_receive_async_API, which led to many timing issues between OTMA C/I and the client.

This enhancement changes OTMA C/I to use the OTMA HOLDQ support for asynchronous (CM0) messages. OTMA C/I specifies it is HOLDQ-capable on the client-bid. The otma_send_async_API was changed to use SendOnly with ACK protocol. The asynchronous output is placed on the HOLDQ. The otma_receive_async_API now uses Resume TPIPE Single to retrieve the output.

These changes to OTMA C/I are internal with the minor effect on customers. Existing OTMA C/I clients see a change during error conditions. The OTMA C/I client that is using asynchronous messaging no longer receives IMS DFS messages when the input is rejected by OTMA. OTMA C/I provides return and reason codes to indicate the specific error. For example, rather than DFS064 for an invalid transaction code, the OTMA C/I clients now receive RC=00000014,RSN=0000001A,0000001D,00000000,00000002. RC=x’14’ indicates OTMA NAK codes, RSN=x’1A’ indicates message cancelled, x’1D’ indicates SMB not found (invalid transaction code).

This enhancement does not change the synchronous Send-then-Commit (CM1) flows.

3.1.7 OTMA client type

IMS OTMA clients (for example, IMS Connect, WebSphere MQ, and OTMA C/I) each has unique processing considerations. IMS Connect is HOLDQ-capable and supports the Resume TPIPE protocol. WebSphere MQ supports synchronous TPIPEs for Commit Mode 1 (Commit-then-Send) messages. OTMA C/I was enhanced to be HOLDQ-capable. In many cases, the differences in client type affect how OTMA processes client requests and messages. The client initialization process (client-bid) was enhanced so that clients can declare their “type” to OTMA. Knowing this type allows OTMA to optimize certain functions for greater efficiency.

IMS Connect in IMS 13, OTMA C/I, and the internal DFSYICAL for Synchronous Program Switch identify their client type to OTMA. IMS OTMA Destination Routing Descriptors can be used to create OTMA clients (TMEMBER) for asynchronous output. If the TMEMBER did not previously exist, the client type is set for TYPE=IMSCON or TYPE=MQSERIES.

3.1.8 Enhancements to OTMA destination descriptors for asynchronous callout

The destination descriptor for IMS asynchronous callout was expanded to introduce a new type, MQSERIES, with which customers can define message switch destinations (from ALT IOPCB) as WebSphere MQ. This new type of descriptor for WebSphere MQ externalizes the definitions in the DFSYDTx member of IMS.PROCLIB rather than forcing customers to code OTMA routing exits (DFSYPRX0 & DFSYDRU0). In all, 13 new parameters for this new descriptor are introduced to prepare the MQMD structure in the OTMA user data prefix.

Dynamic addition, update, or deletion of the destination descriptors for WebSphere MQ can be performed by using the following type-2 commands:

- **`CREATE OTMADESC`**
- **`UPDATE OTMADESC`**
- **`DELETE OTMADESC`**

Figure 3-5 shows the CREATE OTMADESC new TYPE(MQSERIES) option.

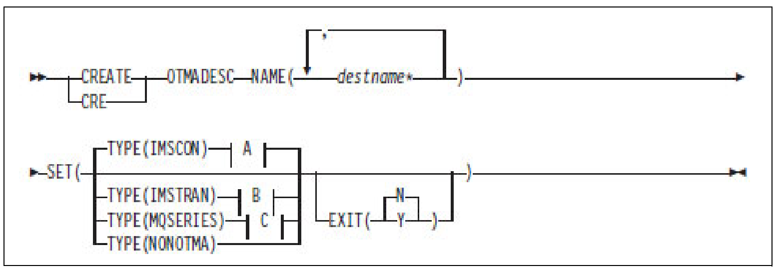


Figure 3-5 CREATE OTMADESC new TYPE(MQSERIES) option

Figure 3-6 shows the type-2 MQSERIES descriptors.

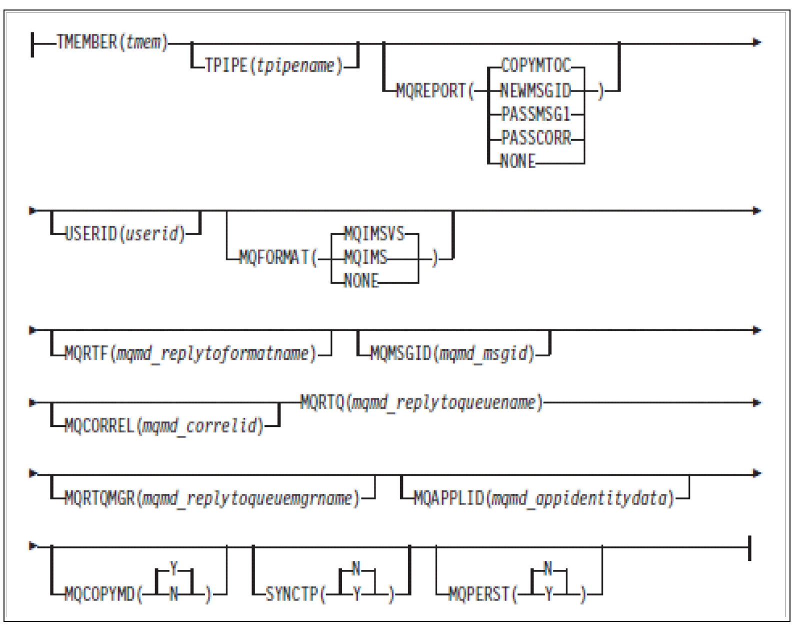


Figure 3-6 C (TYPE=MQSERIES)

Changes that are made to the descriptors by using the type-2 commands are recovered across warm and emergency restarts. However, for IMS cold starts or **/ERE COLDCOMM** -restart, it is recommended that you make the corresponding changes in IMS PROCLIB.

The new destination descriptor support for WebSphere MQ destinations cannot be used for synchronous callout ICAL calls.

The following new TYPE=MQSERIES OTMA descriptors for WebSphere MQ are available with all the parameters that you can define in IMS PROCLIB member DFSYDTx:

```
D MQ*    TYPE=MQSERIES  TMEMBER=VC7  TPIPE=MQTPIPE
D MQ*    USERID=USRT001  EXIT=NO  SYNCTP=YES
D MQ*    MQRTQMGR=VC7  MQRTQ=OTMA.FROM.IMS
D MQ*    MQREPORT=NONE  MQFORMAT=MQIMS  MQPERST=YES
D MQ*    MQMSGID=MSG456789012345678901234
D MQ*    MQCORREL=COR4567890123456789O1234
D MQ*    MQAPPLID=APP45678901234567890123456789012
D MQ*    MQRTF=MQIMSVS  MQCOPYMD=NO
```

The TMEMBER, MQRTQMGR, and MQRTQ parameters are required. The descriptor removes the need to code the OTMA exits, DFSYPRX0, and DFSYDRU0, but they still can be used in addition to the descriptor. The parameter EXIT={YES | NO} specifies whether the OTMA exits are to be called.

A new parameter EXIT= is added to the destination descriptor. When EXIT=YES is specified and OTMA routing exits (DFSYPRX0 and DFSYDRU0) are found in the IMS, the exits can override the routing information that is specified in the descriptor. Exits are passed the descriptor information, and they can accept descriptors routing information or provide override information. Other IMS exits, such as DFSYIOE0, are not affected by this new parameter. The default setting of EXIT is NO, which means that the OTMA routing exits, if found, cannot be called to override the routing information in the descriptor. This parameter is applicable to all destination descriptors (not only TYPE=MQSERIES).

The EXIT= mechanism for providing overrides is simpler than the previous OTMAMD specification, which was a way to define whether the member override function in the DFSYPRX0 user exit is started when a transaction was started from an OTMA client for ALTPCB messages. OTMAMD (N) meant that for transactions that are started from a non-OTMA LTERM, the 16-byte member override field of the DFSYPRX0 parameter list might be used to specify an OTMA client member name, but not for transactions originally started from an OTMA client. OTMAMD (Y) meant that the member override field of the user exit parameter list can be used for OTMA and non-OTMA started transactions.

Figure 3-7 on page 80 shows an example of OTMA descriptor for WebSphere MQ.

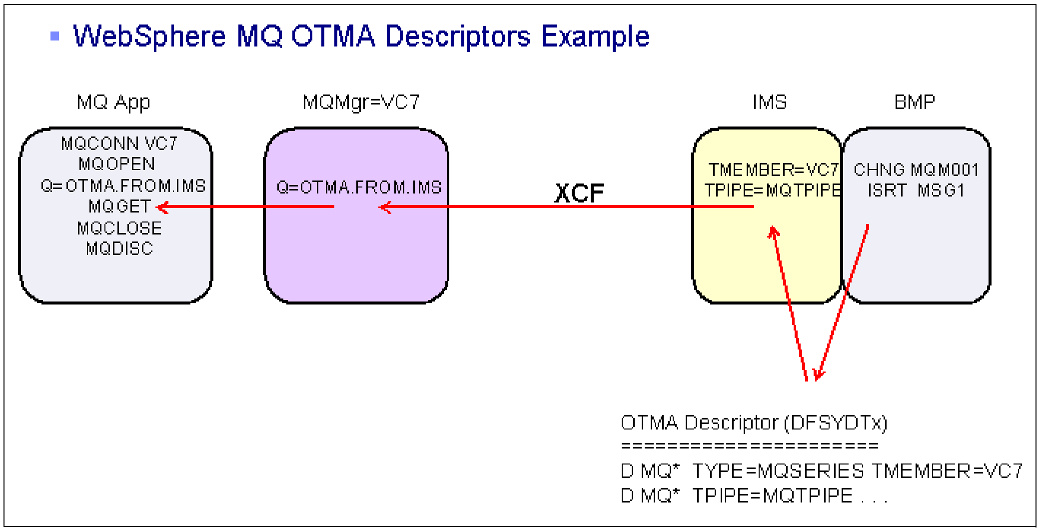


Figure 3-7 Example of WebSphere MQ OTMA descriptor

Changes to the DFSYPRX0 and DFSYDRU0 exits parameter list

A new flag, new input field, and new return code were added to the user exit parameter list in the DFSYPRX0 exit routine. The address at +88 points to a location that stores the routing information that is defined in the descriptor for WebSphere MQ and IMS Connect. If the destination name matches a non-OTMA destination in the descriptor or it does not match any entry in the OTMA destination descriptor, this new field is zero. For the destination as IMS Connect, see the TMAMICON_DESCRIPTOR DSECT mapping for the detailed routing information. For the destination as WebSphere MQ, see the TMAMMQS_DESCRIPTOR DSECT mapping for the detailed routing information.

Similarly, DFSYDRU0 added a new flag, new input field, and a new return code to the user exit parameter list. The address at +100 points to a location that stores the routing information that is defined in the descriptor for WebSphere MQ and IMS Connect. If the destination name matches a non-OTMA destination in the descriptor or it does not match any entry in the OTMA destination descriptor, this new field is zero.

Figure 3-8 on page 81 shows the contents of the parameter lists that are passed to DFSYPRX0 and DFSYDRU0. The changes with IMS 11 are highlighted in red.

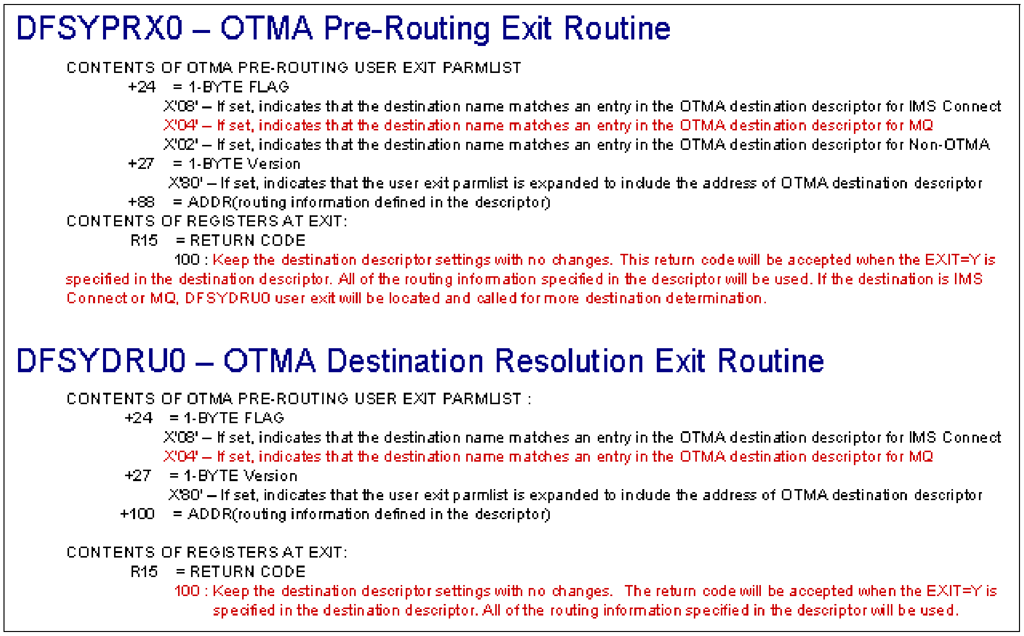


Figure 3-8 DFSYPRX0 and DFSYDRU0 parameter lists

3.1.9 Small programming enhancements

There are some OTMA enhancements that are made available to IMS 11 and IMS 12 as small programming enhancements (SPE). The following SPEs are included in the base code of IMS 13:

- IMS 12 Synchronous Callout SendOnly Ack SPE
- OTMA MAXTP Enhancements
- OTMA Transaction Expiration and Shared Queues SPE

We provide only a short description of these SPEs with the APAR and PTF numbers in this section. For more information, see *IMS Integration and Connectivity Across the Enterprise,* SG24-8174.

IMS 12 Synchronous Callout SendOnly Ack SPE

The Synchronous Callout SendOnly ACK SPE was delivered to address IMS 12 environments and is part of base IMS 13. This function provides a new call for remote servers to use when a reply is sent to an IMS synchronous callout (ICAL) interaction.

With this function, synchronous callout servers can request and receive an indication of response delivery to the IMS ICAL application so they can commit or abort their updated resources. To enable the function, the following APARs must be applied (one to IMS OTMA and the other to IMS Connect):

- IMS Connect 12: APAR PM39569 (PTF UK74666)
- IMS OTMA 12: APAR PM39562 (PTF UK74653)

Figure 3-9 on page 82 shows an example of the callout process, including a request for SendOnly ACK.

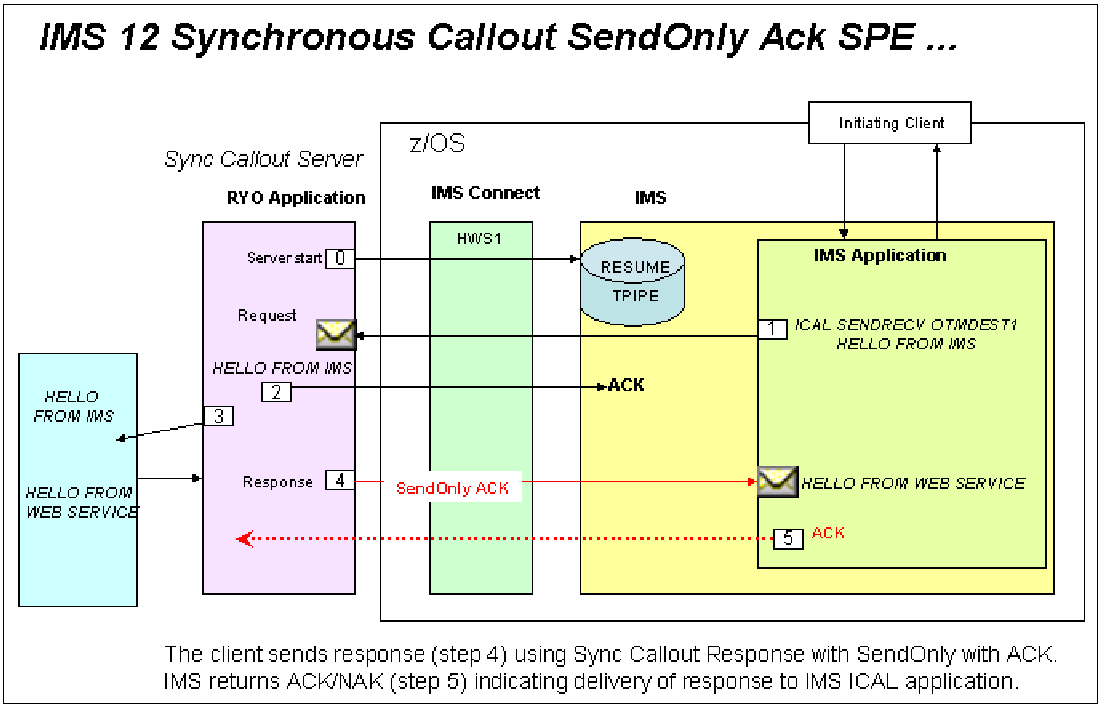


Figure 3-9 Callout process including a request for SendOnly ACK

For a synchronous callout server application to use this new function, another receive for the ACK (acknowledgment) or NAK (negative acknowledgment or rejection) must be coded. This implementation does not affect or require any changes to the IMS application that issues the ICAL. The ACK/NAK is sent by OTMA.

To minimize the possible performance effect of coding another receive, the server can design the interaction to separate the request retrieval threads from the processing and response threads. Different threads can use different sockets.

There is no requirement to use the new function.

OTMA MAXTP enhancements

A TPIPE is a logical connection between IMS and the OTMA client, through which all input and output for a specific client or client ID is routed and queued. OTMA creates TPIPEs dynamically at the request of the OTMA client. Each TPIPE includes extended CSA storage and extended private storage that is associated with it. When there are too many TPIPEs in the IMS system, it can be a storage concern. To limit the growth of OTMA TPIPE creation in the system, the optional parameter (MAXTP) in the OTMA client descriptor was introduced to set a TPIPE limit for an OTMA member. After this parameter is defined for an OTMA member, OTMA begins monitoring requests for TPIPE creation.

The following base support for OTMA MAXTP was introduced with APARs to IMS 11, and IMS 12:

- IMS 11: APAR PM14510 (PTFs UK65904/UK65905)
- IMS 12: APAR PM33681 (PTFs UK75918/UK75919)

The following enhancements to this base support are provided with the IMS 11 OTMA MAXTP SPE, which includes IMS 11 and IMS 12 environments:

- IMS 11: APAR PM71035 (PTFs UK91360/UK91361)
- IMS 12: APAR PM74457 (PTFs UK97118/UK97119)

All of these capabilities are part of the IMS 13 base.

OTMA Transaction Expiration and Shared Queues SPE

OTMA Transaction Expiration and Shared Queues SPE provides the following enhancements:

- Options when transaction expiration occurs at application GU time
- Option to suppress or display symptom dumps and DFS554A messages
- Option to return input message instead of DFS3688I
- Improved routing capability of Shared Queues back-end ALTPCB output
- Support generic or wildcard character at the end of TPIPE name for /DIS TMEMBER TPIPE command

The SPE is included in IMS 13 base code and made available to previous versions via the following APAR/PTFs:

- IMS 10: PM05985 (UK75413/UK75414)
- IMS 11: PM05984 (UK74312/UK74313)
- IMS 12: PM46829 (UK75415/UK75416)

3.2 Non-discardable Message Exit (DFSNDMX0) enhancements

Based on the customer requirement, IMS 13 includes a new function for the non-discardable messages exit routine (DFSNDMX0). The new function is to PSTOP an abending transaction. PSTOP allows queuing to continue, but stops the scheduling.

Four new enhancements for the transaction status flag are provided that enable the DFSNDMX0 exit routine to perform one of the following actions:

- Stop the scheduling of messages that contain a specific transaction code:

  Like **`/PSTOP TRAN`** command
- Stop input messages for a particular transaction code:

  Like **`/PURGE TRAN`** command
- Stop a particular transaction:

  Like **`/STOP TRAN`** command
- Start a particular transaction:

  Like **`/START TRAN`** command

The benefit of this enhancement is to provide more flexibility in the DFSNDMX0 Non-discardable Messages exit routine to better handle abending transactions in a high availability environment.

This enhancement also is made available for IMS 11 and 12 via PTFs. The following APAR and PTF numbers are used:

- IMS 11 APAR PM67950 (PTF UK81040)
- IMS 12 APAR PM45943 (PTF UK71989)

3.3 Multiple Systems Coupling link increased maximums

In IMS Version 13, the maximum number of logical and physical links that you can define for Multiple Systems Coupling (MSC) is increased.

For logical links, the maximum number of MSLINK macros that you can define is now 999 or in IMS systems that are defined as large systems by the LGEN parameter in the IMSCTRL macro, 676.

For physical links, the maximum number of MSPLINK macros that you can define is now 999 or in IMS systems that are defined as large systems by the LGEN parameter in the IMSCTRL macro, 676.

Also, the maximum value of the SESSIONS keyword in the MSPLINK macro is increased to 999.

This enhancement requires APAR PM90983 (PTF UK97740).

3.4 Intersystem Communication over TCP/IP

By using Intersystem Communication (ISC), the IMS Transaction Manager can be connected to other IMS systems, to IBM CICS® Transaction Server for z/OS, and to user-written software.

In IMS Version 13, the ISC function is enhanced to support TCP/IP connections between IMS and IBM CICS Transaction Server for z/OS. The TCP/IP support for ISC is provided in IMS by IMS Connect. ISC support for TCP/IP connections provides a new strategic alternative to the connection support that is provided by the Virtual Telecommunications Access Method (VTAM).

ISC VTAM (SNA LU 6.1) connections continue to be supported in IMS 13 with the new ISC TCP/IP capability.

To communicate with CICS over an ISC TCP/IP connection, IMS uses the private CICS IP interconnectivity (IPIC) protocol for session control and data flow. However, applications programs are not sensitive to the use of IPIC or the use of ISC TCP/IP or ISC VTAM. Unless your application programs use functions that are not supported by ISC TCP/IP, no changes to the existing application programs on either side of the connection are required.

VTAM and TCP/IP connections can coexist.

3.4.1 ISC over TCP/IP overview

IMS 13 enhances Intersystem Communication (ISC) connectivity to include the TCP/IP protocol. This capability applies to connections between IMS and Customer Information Control System (CICS) systems.

This new support provides a strategic protocol alternative to SNA/VTAM environments that want to implement an all inclusive TCP/IP network environment.

At a high level, the ISC support by using TCP/IP protocols can be defined in IMS environments statically and dynamically (ETO).

It uses the Structured Call Interface (SCI) of the Common Service Layer (CSL) between IMS and IMS Connect to use the existing IMS Connect TCP/IP support and requires Operation Manager (OM).

The ISC communication over TCP/IP networks supports the following static and dynamic terminals:

- Static terminal definitions:
  - SYSGEN stage 1 TYPE, TERMINAL, SUBPOOL macros
  - DFSDCxxx PROCLIB member
- Dynamic terminal specification

  Log on descriptors in the DFSDSCMx or DFSDSCTy PROCLIB member
- ISC uses IMS Connect enhancements to the HWSCFGxx configuration member:
  - No IMS Connect user message exit changes
  - ISC over TCP/IP has several IVP and Syntax Checker enhancements

3.4.2 ISC topologies

The following ISC VTAM (SNA LU 6.1) connections continue to be supported in IMS 13 with the new ISC TCP/IP capability:

- Connectivity to CICS using TCP/IP requires CICS 5.1:
  - IMS Connect provides the TCP/IP communication to CICS from IMS
  - Uses new protocols
- VTAM and TCP/IP connections can coexist in both environments.

Figure 3-10 on page 86 shows an IMS and CICS system supporting VTAM and TCP/IP connections.

For ISC TCP/IP, IMS communicates with IMS Connect via the SCI. IMS Connect communicates with CICS via TCP/IP network.

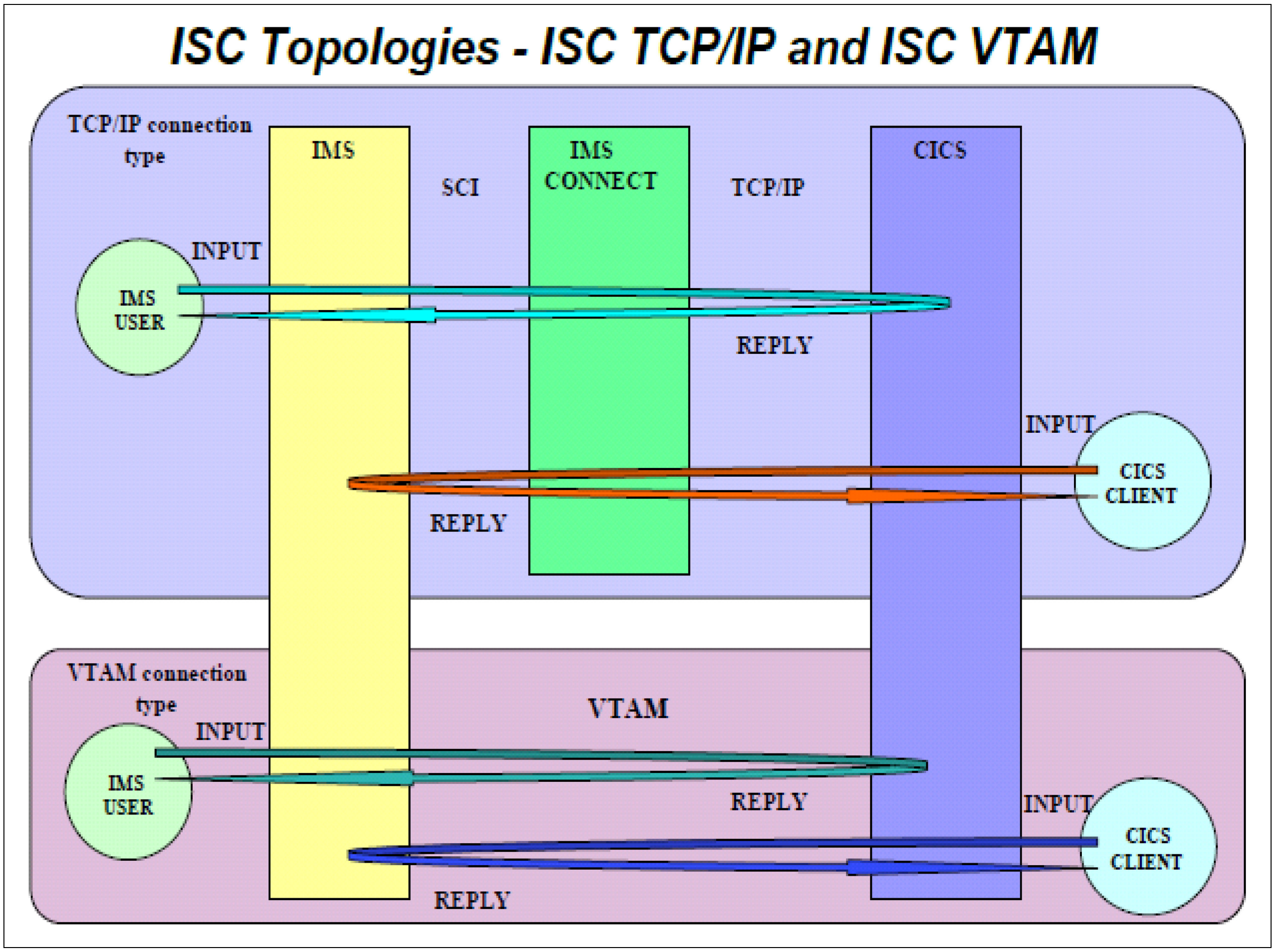


Figure 3-10 IMS and CICS system supporting VTAM and TCP/IP connections

3.4.3 Functionality

Connectivity between CICS and IMS supports only the START/RETRIEVE (asynchronous processing support) in IMS 13, which means that when CICS is the front-end system, the following results are observed:

- IMS response mode transactions are not supported
- IMS conversational transactions are not supported
- CICS application program cannot use SEND/RECEIVE (synchronous) requests

When IMS is the front-end system, the following results are observed:

- Transaction support is the same for ISC VTAM and ISC TCP/IP
- Transaction flow is always asynchronous

For IMS programs and CICS START/RETRIEVE programs, no changes on either side of the connection is required.

Table 3-1 lists the supported or restricted functionality of the ISC TCP/IP support.

Table 3-1 Supported or restricted functionality of ISC TCP/IP support

Functions and transactions	Existing in LU6.1	Supported in TCP/IP
CICS transaction – START/RETRIEVE	Yes	Yes
CICS transaction – SEND(INVITE)/RECV	Yes	No
CICS transaction – SEND(LAST)/RECV	Yes	No
IMS non-response mode transaction	Yes	Yes
IMS response mode transaction (including FP)	Yes	No
IMS conversational mode transaction	Yes	No
IMS recoverable transaction	Yes	Yes
IMS unrecoverable transaction	Yes	Yes
IMS message switch	Yes	Yes
IMS operator command	Yes	Yes
Dynamic terminal	Yes	Yes
Static terminal	Yes	Yes
Front-End Switch (FES)	Yes	No
Message Format Service (MFS)	Yes	No
IMSplex Terminal Management (STM)	Yes	No
VTAM Generic Resources (VGR)	Yes	No
Extended Recovery Facility (XRF)	Yes	No

Flow of a CICS transaction from IMS

Figure 3-11 shows the flow when a CICS transaction is started from IMS by using an ISC TCP/IP connection.

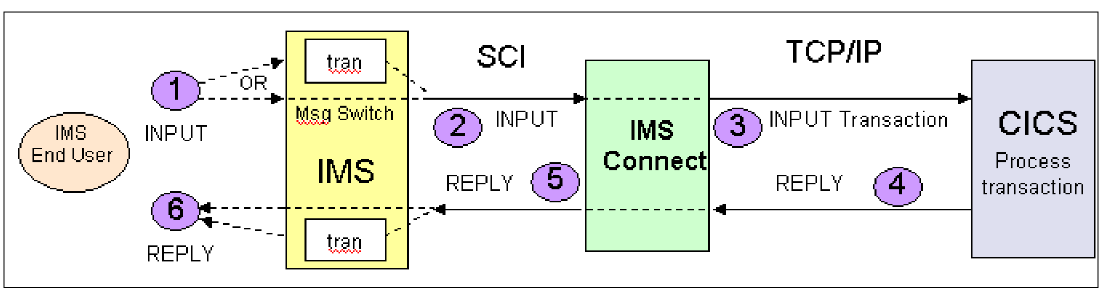


Figure 3-11 Flow when a CICS transaction is started from IMS

The following steps are shown in Figure 3-11:

1. An IMS user sends in a message switch that is to be sent to CICS or an input message to an IMS transaction that issues an ISRT ALTPCB for CICS.
2. IMS uses an active ISC TCP/IP connection to pass the message to IMS Connect by using SCI.
3. IMS Connect uses an active socket connection to send the message to CICS.

4. CICS receives the message as a transaction from TCP/IP network, processes it, and uses an active ISC TCP/IP connection to send the transaction reply to IMS.
5. IMS Connect receives the transaction reply from TCP/IP network and passes it to IMS over an active ISC TCP/IP connection by using SCI.
6. The reply can start an IMS transaction or be a reply that is targeted for an IMS user.

Flow of an IMS transaction from CICS

Figure 3-12 shows the flow when an IMS transaction is started from CICS by using an ISC TCP/IP connection.

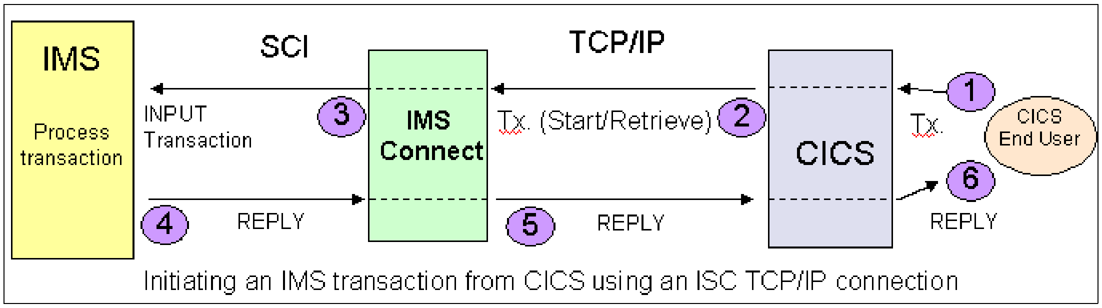


Figure 3-12 Flow when an IMS transaction is started from CICS

The following steps are shown in Figure 3-12:

1. CICS client enters a transaction request.
2. CICS uses an active ISC TCP/IP connection to send the transaction to IMS.
3. IMS Connect receives the transaction from TCP/IP network and uses an active ISC TCP/IP connection to pass the message to IMS by using SCI.
4. IMS processes the transaction and uses an active ISC TCP/IP connection by using SCI to pass the transaction reply to IMS Connect.
5. IMS Connect uses an active socket connection to send transaction reply to CICS.
6. CICS sends the reply to the client.

3.4.4 ISC parallel sessions

Parallel sessions are supported in ISC TCP/IP connections as they are in VTAM.

IMS application supports session (subpool) levels in ISC VTAM and ISC TCP/IP. CICS application supports session levels in ISC VTAM and ISC TCP/IP.

IMS to CICS VTAM communications

Figure 3-13 on page 89 shows an example of a set of ISC VTAM parallel sessions between IMS and CICS.

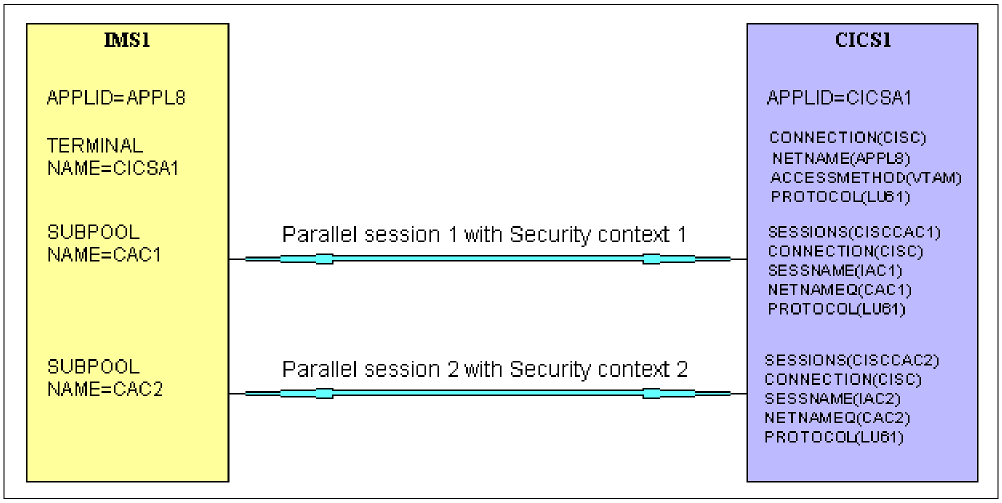


Figure 3-13 Set of ISC VTAM sessions between IMS and CICS

IMS to CICS TCP/IP communications

Figure 3-14 shows an example of a set of ISC TCP/IP parallel sessions between IMS and CICS.

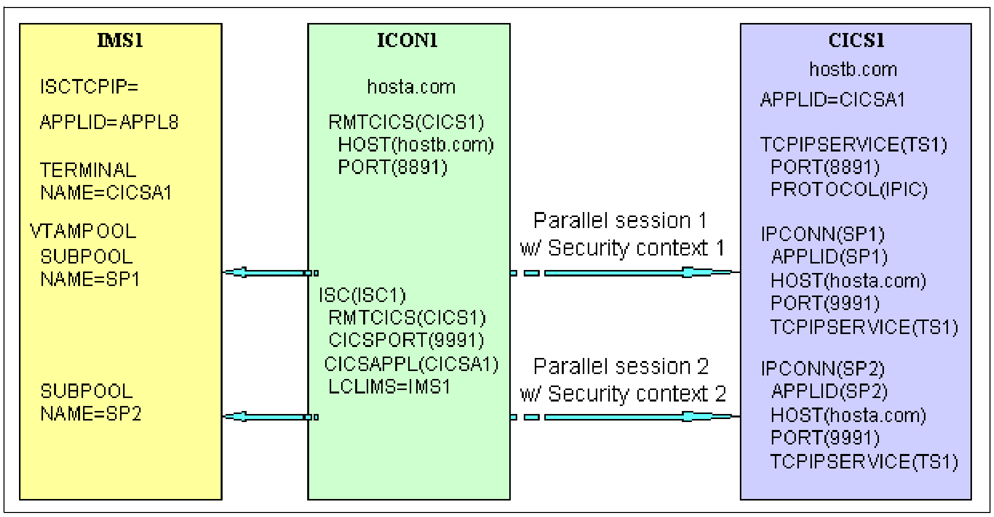


Figure 3-14 Set of ISC TCP/IP sessions between IMS and CICS

3.4.5 ISC TCP/IP support: IMS definitions

In IMS 13, dynamic Extended Terminal Option (ETO) and static ISC terminals can use the new TCP/IP support.

Static terminal support requires a new parameter, ISCTCPIP, that is specified in the DFSDCxxx member of the IMS PROCLIB data set. Dynamic/ETO ISC terminals use logon descriptors that can be specified in the DFSDSCMx or the DFSDSCTy members of the IMS.PROCLIB data set. The use of static versus dynamic ISC TCP/IP terminals provides equivalency in many respects; for example, connections and the status can be started or stopped via IMS commands and input and output messages can be transported by using either terminal type. Where they differ is that dynamic terminals do not require a system definition or a cold start of IMS, which provides much greater flexibility than static ISC terminals.

Static terminal support

IMS enhances the DFSDCxxx PROCLIB member to provide static ISC TCP/IP terminal support for parallel sessions. Static ISC TCP/IP terminals do not support single ISC sessions. Because ISC terminals use VTAM support by default, the ISCTCPIP parameter in the DFSDCxxx PROCLIB member must be defined to enable TCP/IP for static LU 6.1 ISC terminals.

An example is shown in Figure 3-15.

```
DFSDCxxx:
ISCTCPIP=(CICSA1,ICON1)
Where CICSA1 is defined in the stage1 definition as a UNITYPE=LUTYPE6:

Stage 1 sysgen:
 TYPE  UNITYPE=LUTYPE6
     TERMINAL  NAME=CICSA1,SESSION=3...
VTAMPOOL
    SUBPOOL  NAME=SSN1,MSGDEL=SYSINFO
            NAME  SSNA1,COMPT=2,ICOMPT=1
```

Figure 3-15 Matching terminal definitions

ISCTCPIP sets a static LU 6.1 ISC terminal to use TCP/IP communication protocols. The nodename value is the name of an LU 6.1 ISC terminal, as specified on the NAME parameter of the TERMINAL macro that is part of the terminal system definition. The imsconnect_name value is the IMSplex name of a local IMS Connect instance that manages TCP/IP communications for this IMS. The value must match the name that is specified on the MEMBER parameter of the ISC statement or the IMSplex statement in the IMS Connect configuration PROCLIB member (HWSCFGxx).

When you are switching an existing terminal from VTAM to TCP/IP, ensure that UNITTYPE=LUTYPE6 is specified on the TYPE macro that defines the ISC TCP/IP terminal. If UNITTYPE=LUTYPE6 is not specified, an error message is issued when the ISCTCPIP parameter is processed in the DFSDCxxx member and TCP/IP is not enabled for the terminal. Other existing parameters in the TYPE macros continue to apply to the new support.

The EDIT parameter can continue to be used to define any user-provided physical terminal output edit routine or physical terminal input edit routine. For OPTIONS: AUTOSIGN indicates that IMS automatically signs on to static terminals. For ISC terminals the SUBPOOL name is the user ID. NOAUTSGN indicates that IMS does not automatically sign on to static terminals (this is the default). The AUTOSIGN and NOAUTSGN definitions on the TERMINAL macro override those on the TYPE.

For the TERMINAL macro, the following existing keywords apply to the new TCP/IP capability include:

- COMPTn
- EDIT=
- FPBUF=
- MSGDEL=
- OUTBUF=
- SEGSIZE=
- SESSION=
- OPTIONS

The following OPTIONS are applicable to this support:

- AUTOSIGN|NOAUTOSIGN
- FORCRESP|NORESP|TRANRESP
- NOMTOMSG|MTOMSG
- NOPNDST|OPNDST

Although the ISC TCP/IP support does not use VTAMPOOL, there must be at least one VTAMPOOL statement that is defined for parallel session support and one or more subsequent SUBPOOL macros, each of which can be followed by one or more NAME macros (if at least one SUBPOOL statement has an associated NAME statement), as shown in Example 3-2.

Example 3-2 VTAM SUBPOOL definitions

```
TYPE  UNITYPE=LUTYPE6
TERMINAL  NAME=CICSA1,MSGDEL=SYSINFO,EDIT=(NO,NO),      X
     SESSION=3,OPTIONS=NOMTOMSG,COMPT1=(SINGLE1,VLVB),  X
     COMPT2=(SINGLE2,VLVB), COMPT3=MULT1
     VTAMPOOL
     SUBPOOL  NAME=SSN1,MSGDEL=SYSINFO
        NAME  SSNA1,COMPT=2,ICOMPT=1
     SUBPOOL  NAME=SSN2
        NAME  SSNA2
     SUBPOOL  NAME=SSN3
        NAME  SSNA3
```

Dynamic terminal support

For dynamic/ETO terminals, a new logon descriptor can be added to the IMS DFSDSCMx or DFSDSCTy PROCLIB member. At least one logon descriptor must exist for the dynamic ISC TCP/IP support and a unique logon descriptor is required for each ISC TCP/IP connection. A default logon descriptor name is DFSLU6T. As a reminder, IMS optionally produces ETO descriptors during system definition, which places them in DFSDCMx by specifying the ETOFEAT keyword in the IMSCTRL macro.

Subsequent system definitions of the same stage 1 input deck override the DFSDSCMx members. If TSO or a z/OS utility is used to create descriptors, these are placed in member DFSDSCTy, which avoids loss when member DFSDSCMx is replaced. TSO or a z/OS utility can also update descriptors in DFSDSCMx that were created during system definition.

With the ETOFEAT enabled, stage 1 system definitions for ISC terminals can generate logon descriptors, which are placed in the DFSDSCMx member of PROCLIB. However, you can code logon descriptors directly in the DFSDSCTx member. The descriptors that you can specify in the DFSDSCTy member and the syntax and parameters of those descriptors are the same as the descriptors that can exist in DFSDSCMx. If a descriptor is specified on both members, however, the specifications that are coded in the DFSDSCTy member for the descriptor override the specifications that are coded in the DFSDSCMx member.

The logon descriptor for ISC TCP/IP terminal support is shown in the following example:

```
L LUTYPE61  UNITYPE=ISCTCPIP LCLICON=ICON1
```

The descriptor includes the specification of UNITYPE=ISCTCPIP and the LCLICON keyword, which is applicable only when UNITYPE=ISCTCPIP is specified. For ISC TCP/IP, this keyword identifies the local IMS Connect (IMSCON) member within an IMSplex that this LUTYPE6T terminal connects to via SCI. `Lcl_imsconnect_name` is a 1 - 8 alphanumeric character name.

As with static definitions, the following existing parameters for ISC logon descriptors apply:

- COMPTn
- EDIT=
- OUTBUF=
- SEGSIZE=
- SESSION=
- OPTIONS

The following OPTIONS are applicable to this support:

- FORCRESP/NORESP/TRANRESP
- NOMTOMSG/MTOMSG
- NOPNDST/OPNDST

As in previous releases, USER descriptors are not required and default to the use of DFSUSER. If specified, the following keywords are applicable to the ISC TCP/IP support:

- LTERM
- OPTIONS [see logon descriptor]
- RCVYCONV
- RCVYFP
- RCVYRESP
- RCVYSTSN
- SRMDEF

3.4.6 ISC TCP/IP support: CSL for IMS to IMS Connect definitions

A basic Common Service Layer (CSL) configuration is required to support ISC TCP/IP. This minimum configuration includes the SCI and SCI address space for communication between IMS components, such as IMS to IMS Connect. The OM provides type-2 command support for the environment.

The new ISC TCP/IP support does not impose any change on the existing CSL definition structure in the DFSDFxxx member of IMS PROCLIB.

The IMSPLEX plexname parameter value in DFSDFxxx must match the value that is referenced in the IMS Connect setup definitions for ISC, as shown in Figure 3-16.

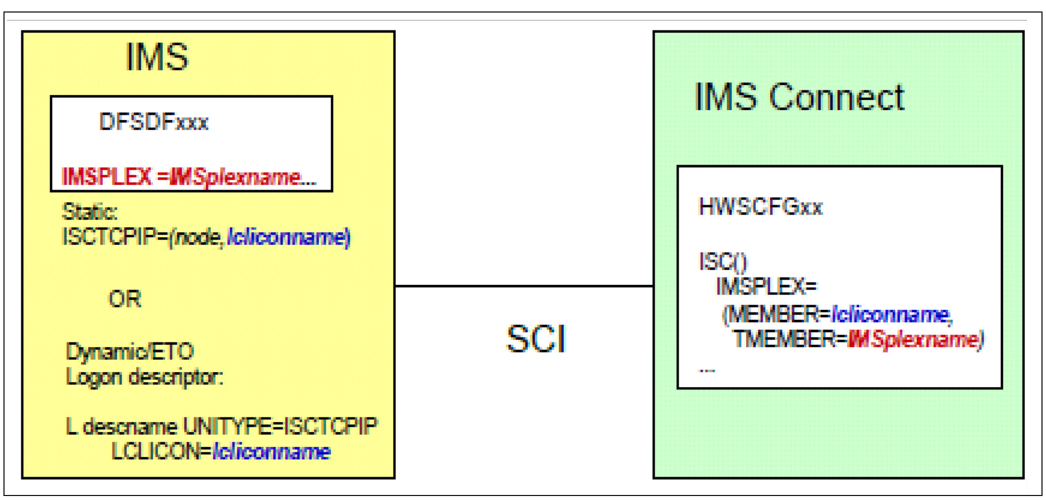


Figure 3-16 IMSPLEX matching definitions

3.4.7 ISC TCP/IP support: IMS Connect definitions

TCP/IP support for ISC is provided by IMS Connect and is defined by coding the appropriate configuration statements in the IMS Connect PROCLIB member, as shown in the following examples:

- New CICSPORT parameter in the existing TC/PIP configuration statement,
- New ISC statement
- RMTCICS statements

The HWSCFGxx TC/PIP statement is enhanced to add the CICSPORT parameter, as shown in Figure 3-17.

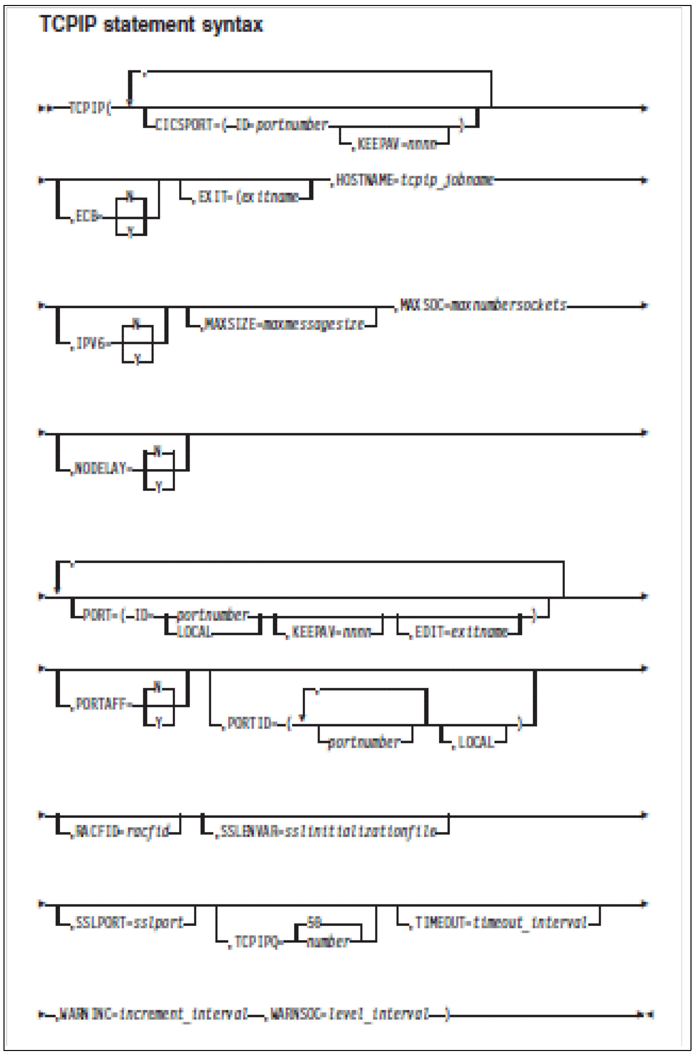


Figure 3-17 TCP/IP syntax in HWSCFGxx

This new parameter identifies a specific CICS port (or ports) to use for the connection. You can define up to 50 TCP/IP ports for an instance of IMS Connect. The combined total for the CICS port (or ports) and all others that might be defined in this IMS Connect instance cannot exceed 50.

As shown in Figure 3-17 on page 93, the new CICSPORT parameter has the following options:

- ID=

  Defines the port (or ports) to use and must be unique for a specific instance of IMS Connect without conflicting with other ports that are selected.

- KEEPAV=

  Specifies the number of seconds for the TCP/IP KeepAlive interval for sockets on this port. TCP/IP accepts a range from 1- 2147460 seconds. If you specify 0, the KeepAlive interval value is bypassed and the setting for the TCP/IP stack is used (this is the default). TCP/IP KeepAlive enables you to detect error situations on inactive sockets.

The ISC statement defines the send path for an ISC link from the local IMS system to a remote CICS system. This ISC statement, with an RMTCICS statement, establishes a one-way send socket path. On the other side of the link, CICS definitions establish the corresponding receive socket for the connection. Additionally, CICS IPCONN definition establishes a send socket on the CICS side, which is matched with a receive socket in IMS. Each ISC parallel session, therefore, establishes two socket connections in IMS Connect. The value that is specified in the LCLIMS parameter and the value that is specified in the NODE parameter must be unique for each ISC statement, as shown in Figure 3-18.

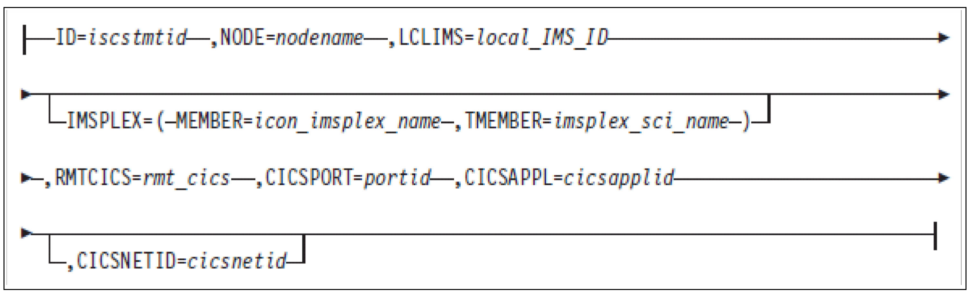


Figure 3-18 ISC statement

The IMSPLEX defines the SCI information for communicating with the local IMS.

The RMTCICS describes the CICS information.The RMTCICS value specifies the remote CICS connection and is the same as the ID= parameter of one of the RMTCICS statements.

ISC links that use separately defined TCP/IP connections to the same CICS subsystem must specify different IDs on the RMTCICS parameter; however, the different RMTCICS statements that each ISC statement references must specify the same host name or IP address and port number.

The following CICS parameters are available:

- CICSPORT=

  Specifies a CICSPORT port ID as defined in the TCP/IP statement. Multiple ISC statements can specify the same CICSPORT port ID if, and only if, they specify the same LCLIMS. However, if these ISC statements specify the same CICSAPPL value, they must have unique CICSPORT values. Otherwise, IMS Connect issues an error message and an abend on start.

- CICSAPPL=

  Equates to the specified APPLID value of the DFHSIT macro definition in the remote CICS. This parameter must be specified whether RMTCICS parameter is also specified; otherwise, IMS Connect issues an error message and an abend on start.

- CICSNETID=

  The VTAM NETID or the UOWNETQL= parameter of the DFHSIT macro definition in the remote CICS subsystem.

The new RMTCICS() statement in HWSCFGxx defines each remote CICS connection that can communicate with this instance of IMS Connect, as shown in Figure 3-19.

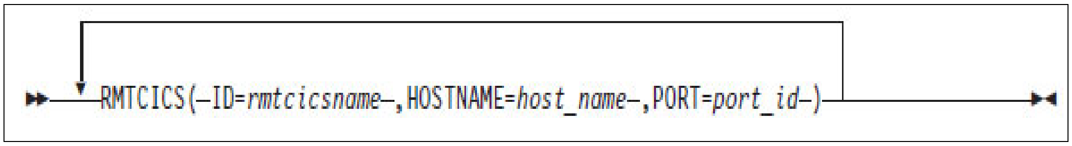


Figure 3-19 RMTCICS statement parameters

The following parameters are used:

- ID=

  Identifies this statement. It matches one or more RMTCICS specifications on the ISC() statement (previous visual).

- HOSTNAME=

  The host name of the remote CICS; for example, CICSA.ibm.com. This parameter can accept up to 60 alphanumeric characters.

- PORT=

  The remote CICS port number and must match a definition in the remote CICS.

- RESVSOC=

  Defines the number of sockets to reserve for use by this connection. This value should be equivalent to the number of parallel sessions. IMS Connect reserves this number of sockets from the maximum number of sockets that is allowed for this instance of IMS Connect, as specified in the MAXSOC= parameter in the TCP/IP statement. If more connections are required for ISC above this RESVSOC count, IMS Connect gets them (if possible) and applies the other sockets towards the max sockets count. The sum of all of the RESVSOC= values that are specified in all the RMTCICS statements must not exceed the value that is specified in the TCP/IP MAXSOC= parameter. If the sum exceeds the MAXSOC value, IMS Connect issues an error message and abend on start.

In the configuration example that is shown in Figure 3-20 on page 96, consider the RMTCICS definition. This example shows that the local IMS Connect instance is defined to use port 8891 for outbound communications with CICS. This configuration is for establishing the SEND sockets from IMS Connect.

```
TCPIP=(HOSTNAME=TCPIP,PORTID(9996,9997,9998,9999),
   RACFID=GOFISHIN,MAXSOC=5000,
   CICSPORT=(ID=9991),CICSPORT=(ID=9992),CICSPORT=(ID=9993),
   CICSPORT=(ID=9994),CICSPORT=(ID=9995),…)

RMTCICS(ID=CICS1,PORT=8891,

HOSTNAME=HOSTB.COM,RESVSOC=10)

ISC(ID=ISC1,NODE=CICSA1,LCLIMS=IMS1,CICSPORT=9991,
RMTCICS=CICS1,CICSAPPL=CICSA1,CICSNETID=CICSNET,
IMSPLEX=(MEMBER=ICON1,TMEMBER=PLEX1))
```

Figure 3-20 HWSCFGXX configuration sample

The ISC statement shows that IMS Connect specified that CICSPORT 9991 is for inbound communication from CICS. This configuration is for establishing the RECV sockets for messages from CICS. Also, the CICSPORT number matches one that was defined in the CICSPORT parameter of the TCP/IP statement.

3.4.8 ISC TCP/IP support: CICS definitions

IPIC or IP interconnectivity requires resources to be defined by using the IPCONN and TCPIPSERVICE statements. A TCPIPSERVICE is the CICS resource that listens for TCP/IP communications from a partner region.

For more information about these CICS parameters, see *CICS Transaction Server for z/OS Version 5 Release 1 Resource Definition*, SC34-2868.

Figure 3-21 on page 97 shows an example of CICS definitions.

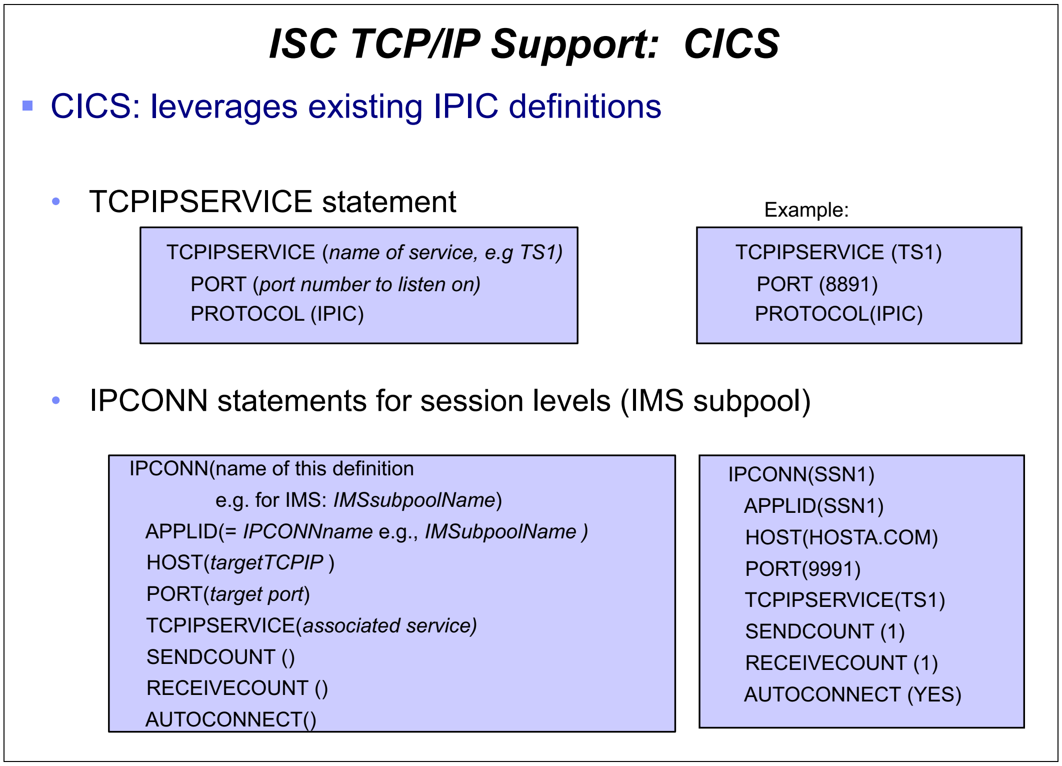


Figure 3-21 IPIC definitions

The following IPIC definitions are used:

- TCPIPSERVICE () is the name of the local TCPIPSERVICE resource where inbound requests are received:
  - PORT

    The port number that CICS listens on for this service.
  - PROTOCOL

    For this type of connection, this value is always IPIC.
- IPCONN ()

  The name of the definition. It matches the following IMS subpool names:
  - APPLID

    Used for the IMS environment, which must match the IPCONN name; for example, IMS subpool name. Because the APPLID value is equal to the IMS subpool name, the connection can be established by IMS or CICS.
  - AUTOCONNECT

    Specifies whether YES | NO sessions are to be established when the IPCONN definition is installed.
  - HOST

    The name of the TCP/IP Stack on the target system MVS image.
  - PORT

    The port that is named on the TCPIPSERVICE resource of the remote system.

– RECEIVECOUNT(1-999)

Specifies the number of sessions that receive incoming requests. The actual number of receive sessions that are used depends also on the number of send sessions that are defined in the remote system; for example, IMS. When the connection is established, these values are exchanged and the lower value is used.

– SENDCOUNT(0-999)

Specifies the number of sessions that send outgoing requests. The actual number of send sessions that are used depends also on the number of receive sessions that are defined in the remote system; for example, IMS. When the connection is established, these values are exchanged and the lower value is used. If 0 is specified, this IPCONN can process incoming work only. It cannot send requests to the connected system and cannot be acquired. A SENDCOUNT value greater than 0 requires PORT to have a numeric value.

CICS support use the following IPIC connectivity capability that exists in CICS:

- Two-socket connections per parallel session: one socket for send and one for receive on each side.
- From IMS to CICS:
  – IMS Connect uses its SEND socket to send a message to CICS.
  – CICS uses it RECEIVE socket to receive the message and uses its definitions to route the message to the CICS destination.
- From CICS to IMS:
  – CICS uses its SEND socket to send a message to IMS Connect
  – IMS Connect uses it RECEIVE socket to receive the message and uses its definition to route the message to the IMS destination

3.4.9 IMS to CICS ISC TCP/IP communications

Figure 3-22 shows the summary of IMS to CICS TCP/IP definitions.

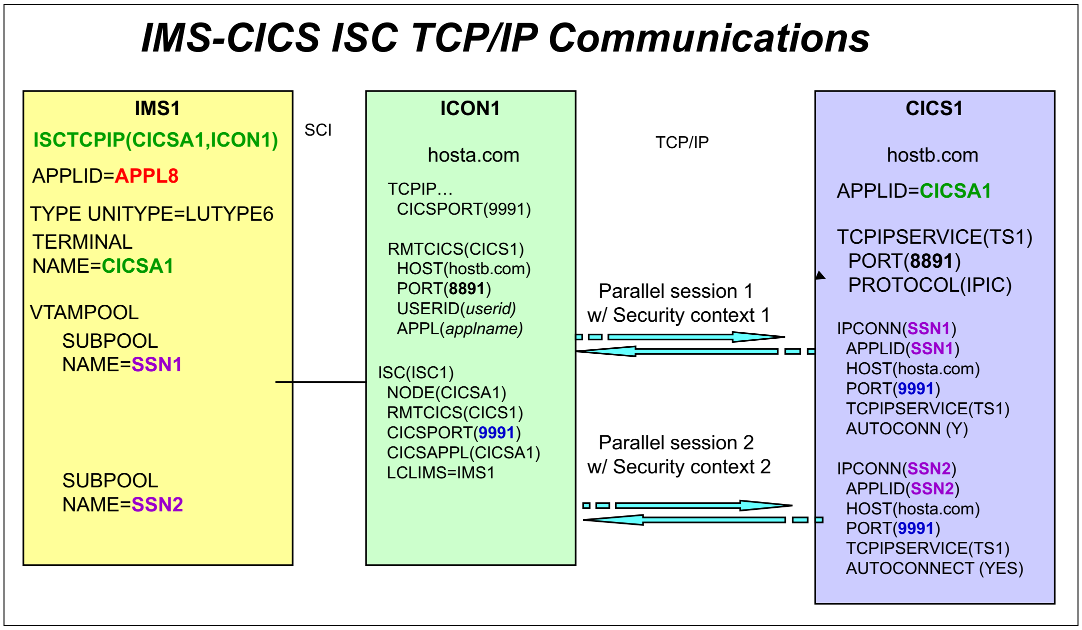


Figure 3-22 Summary of IMS to CICS configuration

On the IMS side, this configuration shows two statically defined parallel sessions and subpools SSN1 and SSN2. Corresponding definitions are provided in CICS with IPCONN statements matching the subpools.

IMS starts the session by using the following /OPNDST command:

```
/OPN NODE CICSA1 USER SSN1
```

Alternatively, CICS can start the session by using the following CEMT (CICS master terminal) transaction:

```
F CICSA1,CEMT S IP(SSN1) ACQ
```

4

Database and Database Recovery Control enhancements

The IMS Version 13 Database Manager includes enhancements to Database Recovery Control (DBRC) and new support for database versioning and dynamic database restructuring.

For more information about DB versioning, see Chapter 6, “Database versioning” on page 181.

This chapter includes the following topics:

- HALDB alter
- DEDB Alter
- Fast Path secondary index enhancement
- DBRC DELETE.LOG INACTIVE and TOTIME commands

4.1 HALDB alter

HALDB Alter is for Full Function HALDB database customers who want to make segment changes without unloading and reloading the database. This feature addresses the challenge of maintaining database availability while changing the structure of a HALDB database.

When a segment change is made to a database description (DBD), an online command can be issued to apply the change to the database. The change is implemented via an option of the HALDB Online Reorganization (OLR) function. Application programs can access the database at the same time the OLR function is changing the structure of the database.

This function provides value to customers by reducing the complexity of making structural changes to a HALDB database, eliminating system downtime, and improving system availability. By using database versioning, this line item can also reduce the cost and risk that are associated with making and coordinating wholesale changes to all application programs when database structure changes occur. In this way, customers can alter their HALDB database structures, even if the database is used by critical application programs that they no longer can change.

4.1.1 HALDB Alter overview

The following structural changes can be applied to an online HALDB with the type-2 **INIT OLREORG** command:

- Adding fields to space at the end of the segment
- Increasing the length of existing segment by adding space at the end of the segment

After a segment change is made to a DBD, an online reorganization command can be issued to start the change to the database. The change is implemented via the **INIT OLR** option of the type-2 HALDB Online Reorganization (OLR) function.

Application programs that use the old database definitions can access the database while the OLR function is altering the structure of the database and until an online change is performed. When a segment change is made to a DBD, an online command can be issued to start the change to the database. The change is implemented via an ALTER option of the type-2 HALDB Online Reorganization (OLR) function. The type-1 **/INIT OLR** command does not support the new HALDB ALTER function.

An Online Change must be completed to activate the DBD change in the online system. Application programs that use the old database definitions can access the database while the OLR function is altering the structure of the database. After an OLC is completed for the DBD, application programs can use the new database definitions to access the database.

HALDB Alter is supported for database types of PHDAM and PHIDAM only. As HALDB Alter relies on the use of type-2 commands, it requires the Common Service Layer (CSL) Structured Call Interface (SCI) and Operation Manager (OM) address spaces to be set up. Another requirement is that the DBRC MINVERS must be set to “13.1” in RECON.

Figure 4-1 shows examples of adding fields to space at the end of the segment. The first example is for adding a field to the beginning of free space at the end of a segment. The second example is for adding a field to the end of free space at the end of a segment. The new field definitions are added to the DBD segment definition.

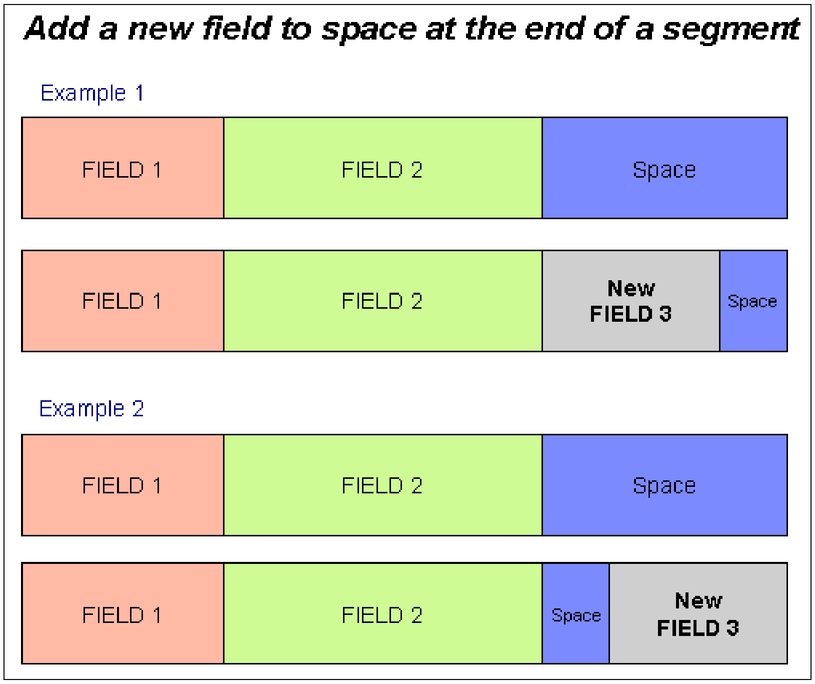


Figure 4-1 Examples of adding new fields to space at the end of the segment

Figure 4-2 on page 104 shows examples for increasing the length of an existing segment. The first example is for increasing the length of an existing segment by adding space at the end. The second example is for increasing the length of an existing segment by adding more free space at end.

The new space is added to the segment size in the DBD definition, new length on BYTES= parm of the SEGM statement.

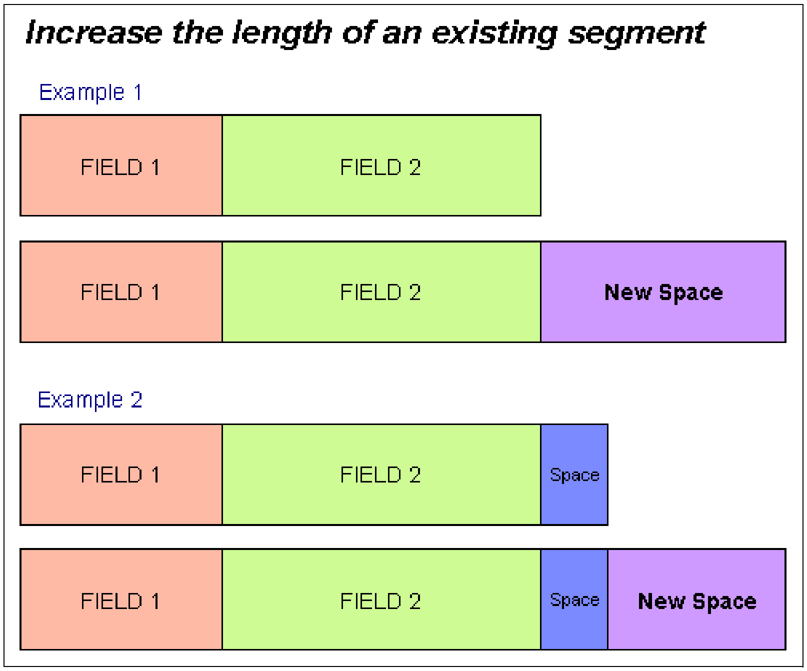


Figure 4-2 Examples for increasing the length of an existing segment

Figure 4-3 shows examples of defining new fields to remap existing fields and adding space in a segment. The first example is for defining two new fields (FIELD 3 and FIELD 4) to overlay/re-map FIELD 2 and defining new FIELD5 in existing free space.

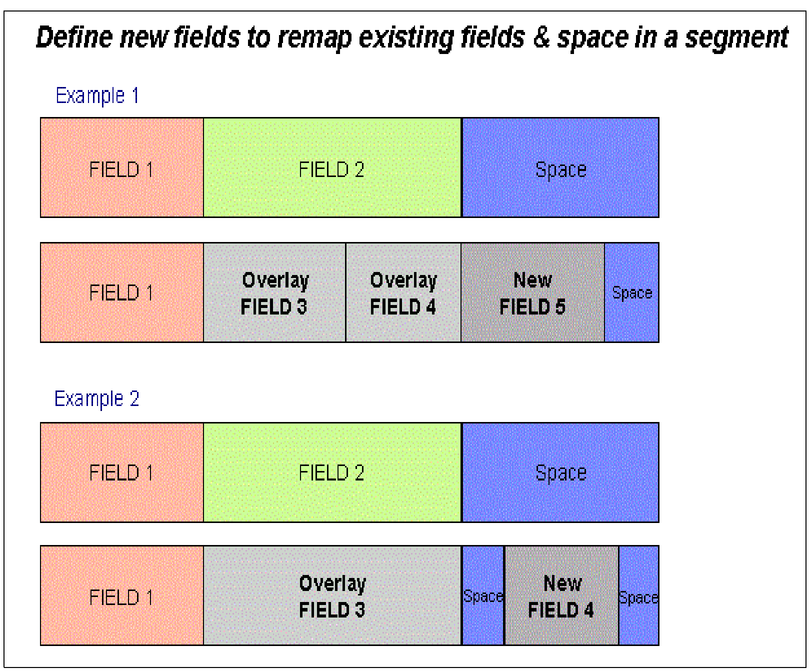


Figure 4-3 Example of defining new fields to remap existing fields and space in a segment

4.1.2 HALDB Alter preparation

Begin the ALTER process by making coding changes to the DBD source. Multiple segment definitions in one DBD can be changed at the same time. You can make the following changes:

- Define new fields in space at the end of segment (or segments): Specify new FIELD statements.
- Increase the length of a segment (or segments): Specify new length in the BYTES= parameter of SEGM statement.

The appropriate DBDGEN and ACBGEN must be run by using the modified source. The ACB member must be generated to an output staging ACBLIB. Modify or code new application programs to use new fields.

The ACBLIB staging library must have a dynamic allocation member. A DFSMDA member can be created for the ACBLIB staging library if one does not exist by using the DFSMDA macros. An example of DFSMDA macros for the staging ACBLIB is shown in Example 4-1.

Example 4-1 DFSMDA macros for the staging ACB library

```
DFSMDA TYPE=INITIAL
DFSMDA TYPE=IMSACB,DSNAME=IMS.STAGING.ACBLIB
DFSMDA TYPE=FINAL
```

During ALTER processing, IMS processes all of the changes that are made in the DBD that are found in the staging ACBLIB for that DB. All segment changes are made at one time.

If you are increasing the size of a segment when you are altering an online HALDB database, you might also need to increase the OSAM block size or VSAM CI size of the output database data set that holds the altered segment.

New block or CI sizes are applied to the output data sets at the start of alter processing, but must be entered in the RECON data set before the **INIT OLREORG OPTION(ALTER)** command is issued.

New block or CI sizes are entered into the RECON by specifying them on the ALTERSZE keyword of the **CHANGE.PART** command or by specifying them in the Change Dataset Groups panel of the HALDB Partition Definition utility (%DFSHALDB). NOALTRSZ keyword can be used to clear block/CI size.

For VSAM data sets, if output data sets for alter processing exist, the output data sets that require a new CI size must be deleted before the alter process is started. Alter processing automatically re-creates the required output data sets with the new CI size. If no ALTERSZE value is specified for a specific VSAM data set group and an output data set exists, the CI size of the output data set is used. If no ALTERSZE value is specified and an output data set does not exist, the CI size of the input data set is used.

For OSAM data sets, if no ALTERSZE value is entered, the BLKSZE of the input data set is used, even if an output data set exists.

When you change a block or CI size, you might also need to change the size of the buffers. If the new block or CI size does not fit into the current buffer subpool, IMS tries to find a larger subpool among the available subpools. If none of the available subpools are large enough to hold the new block or CI size, the output data set fails to open. To check buffer sizes, issue the type-2 command **QUERY POOL TYPE(DBAS)**.

4.1.3 CHANGE.PART command

To increase the OSAM block size or VSAM CI size of the database data sets when you are modifying the structure of a database with the HALDB alter function, you must set ALTERSZE values for each data set group that is changing in each partition record in the RECON.

To set the ALTERSZE values, you can use the DBRC command **CHANGE.PART** or the HALDB Partition Definition utility (%DFSHALDB).

If you use the **CHANGE.PART** command to set the ALTERSZE values, the values must be specified as positional, comma-separated values. The value in the first position applies to the first data set group. The value in the second position applies to the second data set group, and so on.

For example, the ALTERSZE(,4096) keyword sets a new block or CI size for the third data set group, but leaves the sizes unchanged for the first and second data set groups, and for the 4th - 10th data set groups, if they exist.

You can determine the position in which to enter a size for a data set group by looking at the DSGROUP keyword in the SEGM statement that defines the segment that you are altering. DSGROUP=A indicates the first position, DSGROUP=B, the second position, and so on, up to DSGROUP=J, which indicates the 10th position.

ALTERSZE value must be even and no greater than 32 K. New sizes must be stored in the RECON partition record before the **INITIATE OLREORG OPTION(ALTER)** command is issued. New sizes are stored in the RECON partition record until the alter operation is complete and an online change is performed. After online change is performed, new OSAM block sizes are saved in the OSAM BLOCK SIZE field of the RECON partition record.

To correct an ALTERSZE value that is set, replace the incorrect value with the correct value by using the **CHANGE.PART** command or the HALDB Partition Definition utility (%DFSHALDB).

For OSAM data sets, if you change an ALTERSZE value back to the original block size of the input data set, the ALTERSZE value displays as 0 to indicate that the block size is not changing. If all of the ALTERSZE values are restored to the original block sizes of the input data sets, the ALTER SIZE field is omitted from the output of the **LIST.DB** command.

For VSAM data sets, if you change an ALTERSZE value back to the original CI size, the original CI size is displayed. If all of the ALTERSZE values are restored to the original CI sizes of the input data sets, the ALTER SIZE field is displayed with the last values that you entered.

You can clear all of the ALTERSZE values for a partition by specifying **CHANGE.PART PART(name) NOALTRSZ** command. When the NOALTRSZ keyword is used to clear all ALTERSZE values for OSAM and VSAM data sets, the ALTER SIZE field is omitted when the partition record is displayed.

After the command is successfully processed, the block or CI sizes to be used by the alter process are listed under **ALTER BLOCK SIZE** in the RECON record for a partition, which can be displayed by the issuing DBRC command **LIST.DB DBD(partitionname)**.

After the alter size values are corrected, you can start the alter process by issuing the IMS type-2 command **INIT OLREORG NAME(masterdb) OPTION(ALTER)**.

4.1.4 HALDB Alter online process

When the ALTER option is specified, the **INIT OLREORG** command starts a reorganization of an entire HALDB database to apply the database changes to all of the database partitions. During ALTER processing, IMS processes all the changes that are made in the DBD that are found in the staging ACBLIB for that DB. All segment changes are made at one time.

Upon receiving the **INIT OLREORG OPTION(ALTER)** command, an IMS system can alter up to 10 partitions concurrently. Any partitions that cannot be processed immediately are queued internally until they can be altered.

While an IMS system reorganizes and alters a partition, the IMS system has ownership of the partition. The subsystem ID of the IMS system that owns a partition for alter processing is recorded in the OLRIMSID field of the partition record.

In a data-sharing environment, ownership of a partition is granted to the first IMS system that is available to alter the partition. If one IMS system is available to process 10 partitions before any other IMS system becomes available, all 10 partitions are processed by the single IMS system. If partitions are queued for alter processing, ownership of the queued partition is granted to the first IMS system to be altering fewer than 10 partitions concurrently.

As soon as the **INIT OLREORG OPTION(ALTER)** command is received by IMS, DBRC marks every partition in the database with ALTER IN PROGRESS=YES, even those partitions that are queued. While an IMS system is actively altering a partition, the partition record shows OLREORG CURSOR ACTIVE=YES.

The response to the INIT OLRERORG OPTION(ALTER) is shown in Figure 4-4.

```
Response for: INIT OLREORG NAME(C2CSTP) SET(RATE(050)) OPTION(ALTER,NODEL)
Partition MbrName    CC

C2CST01   IMDD       0
C2CST02   IMDD       0
C2CST03   IMDD       0
C2CST04   IMDD       0
C2CST05   IMDD       0
C2CST06   IMDD       0
C2CST07   IMDD       0
C2CST08   IMDD       0
C2CST09   IMDD       0
C2CST10   IMDD       0
C2CST11   IMDD     1E3
C2CST12   IMDD     1E3
C2CST13   IMDD     1E3
C2CST14   IMDD     1E3
C2CST15   IMDD     1E3
C2CST16   IMDD     1E3
C2CST17   IMDD     1E3
C2CST18   IMDD     1E3
C2CST19   IMDD     1E3
C2CST20   IMDD     1E3
```

Figure 4-4 Response to the INIT OLRREORG OPTION(ALTER)

When the **INIT OLREORG OPTION(ALTER)** command is received by IMS, DBRC marks every partition in the database with ALTER IN PROGRESS=YES, even those partitions that are queued. While an IMS system is actively altering a partition, the partition record shows OLREORG CURSOR ACTIVE=YES as shown in Figure 4-5.

```
DB
 DBD=C2CST03   MASTER DB=C2CSTP    IRLMID=*NULL  CHANGE#=3     TYPE=PART
 USID=0000000003  AUTHORIZED USID=0000000003  HARD USID=0000000003
 RECEIVE USID=0000000003  RECEIVE NEEDED USID=0000000000
 DSN PREFIX=USER.IMS13.C2CSTP                         PARTITION ID=00003
 PREVIOUS PARTITION=C2CST02     NEXT PARTITION=C2CST04
 OLRIMSID=**NULL**    ACTIVE DBDS=A-J
 REORG#=00004
 ONLINE REORG STATISTICS:
     OLR BYTES MOVED = 209171
     OLR SEGMENTS MOVED = 1313
  OLR ROOT SEGMENTS MOVED = 101

 FLAGS:                              COUNTERS:
   BACKOUT NEEDED        =OFF          RECOVERY NEEDED COUNT   =0
   READ ONLY             =OFF          IMAGE COPY NEEDED COUNT =0
   PROHIBIT AUTHORIZATION=OFF          AUTHORIZED SUBSYSTEMS   =1
                                       HELD AUTHORIZATION STATE=3
                                       EEQE COUNT              =0
   TRACKING SUSPENDED    =NO           RECEIVE REQUIRED COUNT  =0
   OFR REQUIRED          =NO           OLR ACTIVE HARD COUNT   =0
   PARTITION INIT NEEDED =NO           OLR INACTIVE HARD COUNT =0
   OLREORG CURSOR ACTIVE =NO
   PARTITION DISABLED    =NO
   ONLINE REORG CAPABLE  =YES
   REORG INTENT          =NO
   QUIESCE IN PROGRESS   =NO
   QUIESCE HELD          =NO
   ALTER IN PROGRESS     =YES
   PARTITION ALTERED     =NO
```

Figure 4-5 LIST.DB with ALTER active

After alter processing is complete for a partition, the partition record shows ALTER COMPLETE=YES. The OLR tasks are processed individually by the online system. If one or more of them fail to complete (although the OLR tasks are started by a single command), each partition is processed by a single OLR task. Messages are issued for those tasks that have issues but the other OLR tasks continue to process. The OLR tasks for all the partitions must complete successfully before the ALTER can be completed. If any OLR task fails, the **OLR** command can be re-entered, and IMS tries again the OLR for those partitions that failed. The OLR command must be entered with the master HALDB name and the OPTION(ALTER) as it was before. All partitions that are not complete are tried again.

Only after all partitions in the database have a status of ALTER COMPLETE=YES can you perform online change to activate the new database structure. The online change function resets both the ALTER IN PROGRESS field and the ALTER COMPLETE field to NO. You must issue the **/DBR** command on the HALDB from the online systems before the OLC being started. It is at this point that the applications must be changed to allow for the longer segment lengths.

At a minimum, a recycle of the MPR or CICS regions might be required to bring in the newly compiled program load units. The OLR start and completed messages are written to the DLISAS region of the online system. If the HALDB has more than 10 partitions, 10 OLRs are started and new ones started as others complete. The messages are described in Figure 4-6. When the last of the OLRs complete, a DFS3198I message is issued.

```
DFS3197I HALDB ALTER STARTED FOR NAME=C2CSTP    PARTITION NUMBER=20   IMDD

DFS2970I OLR STARTED FOR NAME=C2CST01  MASTER=C2CSTP   PST=00001 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST02  MASTER=C2CSTP   PST=00002 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST03  MASTER=C2CSTP   PST=00003 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST04  MASTER=C2CSTP   PST=00004 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST05  MASTER=C2CSTP   PST=00005 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST06  MASTER=C2CSTP   PST=00006 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST07  MASTER=C2CSTP   PST=00007 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST08  MASTER=C2CSTP   PST=00008 IMDD
DFS2974I OLR COMPLETED FOR NAME=C2CST03  MASTER=C2CSTP   PST=00003 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST09  MASTER=C2CSTP   PST=00009 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST10  MASTER=C2CSTP   PST=00010 IMDD
DFS2974I OLR COMPLETED FOR NAME=C2CST06  MASTER=C2CSTP   PST=00006 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST11  MASTER=C2CSTP   PST=00003 IMDD
DFS2970I OLR STARTED FOR NAME=C2CST12  MASTER=C2CSTP   PST=00006 IMDD
DFS2974I OLR COMPLETED FOR NAME=C2CST08  MASTER=C2CSTP   PST=00008 IMDD
.....

DFS3198I HALDB ALTER COMPLETED FOR NAME=C2CSTP    PARTITION NUMBER=20
STARTED=20   COMPLETED=20   IMDD
```

Figure 4-6 OLR messages from the DLISAS region

HALDB OLR Alter command

The **INITIATE OLREORG** command is used to start the dynamic structural change processing for HALDB partitions. The structural changes of the HALDB partitions are made in the DBD statement and can be accompanied by the DBVER parameter to identify the version of the database structure change. When the **INITIATE OLREORG** command is run, it reads the new definitions from the staging library and is used to construct the OLR output data set as though the entire database is being reorganized (that is, all of the partitions of a HALDB).

The following keyword parameters are changed or added to the type-2 **INITIATE OLREORG** command:

- NAME()

  This keyword specifies the name of a HALDB master. Unlike the existing INITIATE OLREORG command (which PHDAM or PHIDAM HALDB partition names can be specified), only one HALDB master name can be specified here. You cannot use the wildcard character (*).

- OPTION()

  By using this keyword, you can specify the different options that affect how HALDB online reorganization performs.

- ALTER

  This option specifies that the command starts the database structural change processing. The new database definition is obtained from the staging library, so the new DBD for the database that is specified on the NAME() parameter must be generated into the staging library before the command is run.

Example 4-2 shows an **INITIATE OLREORG** command.

Example 4-2 INITIATE OLREORG command

```
INITIATE OLREORG NAME(masterdb) OPTION(ALTER)
```

Consider the following points regarding the **INITIATE OLREORG** command:

- Starts the dynamic structural change processing for HALDB partitions
- Only one HALDB master database name can be specified per **INIT** command
- IMS reads the new DBD definition from a staging ACBLIB
- Staging ACBLIB is dynamically allocated
- Member information is used to construct the OLR output data set
- Output data sets for all the HALDB partitions are built
- Type-2 **TERM OLREORG** or type-1 **/TERM** command is allowed while database structure change is in progress
- Type-2 **INIT OLR OPTION(ALTER)** command restarts the HALDB ALTER structure process where it left off

You can stop alter processing of a HALDB database before it is complete by issuing the **TERMINATE OLREORG** command or **/TERM** command. You can issue the type-1or type-2 version of the **TERMINATE OLREORG** command; however, only the type-2 version of the command can be issued to multiple IMS systems. The type-1command can be processed only on the IMS that owns the OLR (that is, wherever the partitions are being reorganized).

The **TERMINATE OLREORG** command does not support the specification of the name of a HALDB master database. To stop alter processing for the entire HALDB database, you specify a wildcard character in place of the partition names or you can specify the names of all of the database partitions explicitly. If multiple IMS systems are altering the database, you must use the type-2 **TERMINATE OLREORG** command to stop all of the IMS systems at once or issue the type-1 command separately on each IMS system.

To stop alter processing for one or more partitions of a HALDB database, issue the following command:

```
TERMINATE OLREORG NAME(partnm |*)
```

When alter processing is stopped for a subset of the partitions in the database, alter processing continues for the other partitions that are not contained within the specified subset.

When alter processing is stopped, the data in the partition might be physically stored in the input and output data sets. The output data sets conform to the altered database structure. The input data sets conform to the old database structure. However, where the data is physically stored is not apparent to application programs. Until alter processing completes and online change is performed, application programs can access the data only in the old database structure.

To resume alter processing, issue the **INITIATE OLREORG OPTION(ALTER)** command.

To resume alter processing of a partition on a different IMS system, you can release the ownership of an IMS system by specifying the REL option; for example, **INITIATE OLREORG OPTION(ALTER,REL)**.

HALDB Alter completion by using online change

The IMS online change function is required to enable access to the new structure of a HALDB database after alter processing completes. For application programs that are accessing the altered database, IMS reads the database by using the unaltered DBD and returns the unaltered segment structure until OLC is completed.

Before you start the online change procedure to complete an alter operation, you must stop access to the HALDB database by issuing one of the following commands:

- **/DBR DB HALDB_master_name**
- **UPDATE DB NAME(HALDB_master_name) STOP(ACCESS)**

Note: Do not use the **UPDATE START(QUIESCE)** command.

In addition to activating the ACB members that contain the new database structure, the online change function clears various flags and counters in the RECON data set. Activating the ACB members by a means other than the online change function does not clear the flags and counters automatically.

Until the ACB members are activated and the flags and counters are cleared, the alter procedure is not complete, and the new database structure cannot be used.

Use the member online change function to complete the alter procedure. The member online change function reads directly from the staging ACB library and can process only the specific ACB members that require activation. The local and global online change functions require you to copy the ACB members into the inactive ACB library. They also process the entire ACB library, instead of just the ACB members that contain the new database changes.

Only application programs that use the unaltered database structure can access the database until all of the partitions in the HALDB database are altered and online change is performed. After that process is completed, start access to the altered HALDB database and implement new or modified IMS application programs. The new HALDB database structure can now be used by modified or new application programs that need the new segment fields.

4.1.5 HALDB ALTER example

Figure 4-7 on page 112 shows a walk-through example of HALDB Alter that increases the segment ‘B’ size from 30 to 40 bytes.

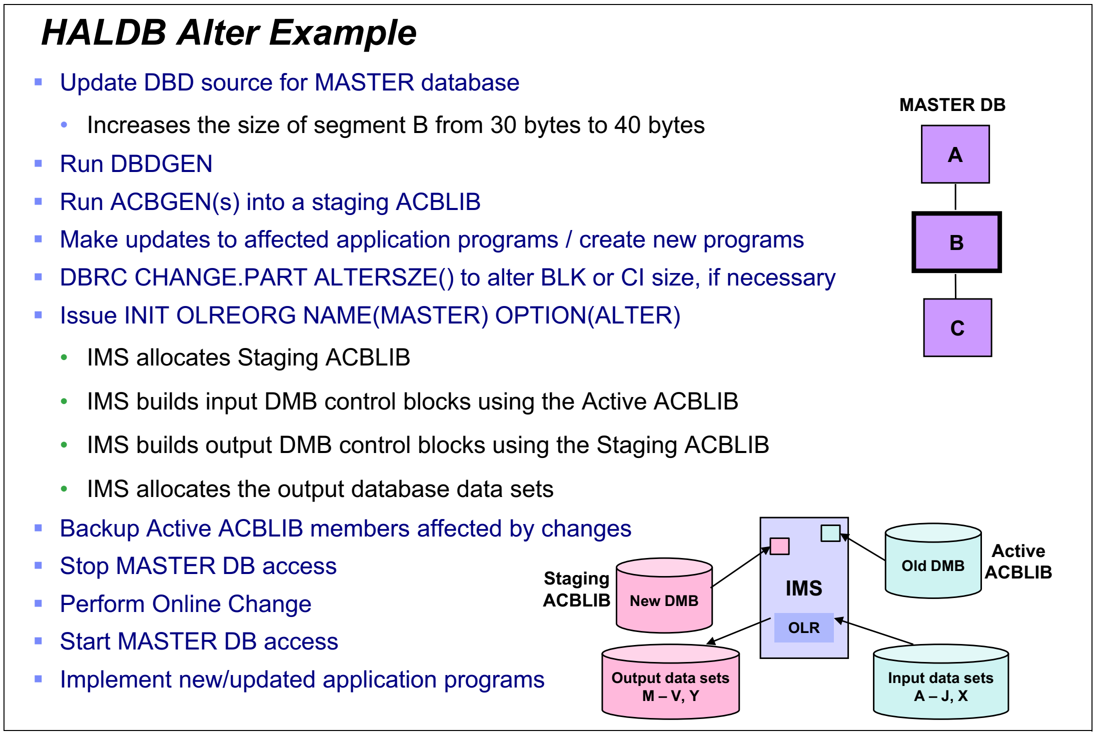


Figure 4-7 Required steps for HALDB Alter

As part of building the input and output DMB control blocks, IMS compares the segments and fields and tracks the changes. Until OLR ALTER and OLC are complete, IMS reads the segment by using the old DMB and returns the unaltered segment structure.

4.1.6 HALDB Alter segment field fill values

Segment fill values are based on the field type. For fields added at the end of a segment

The contents of the fields that are added at the end of a segment depend on the following TYPE definition:

- ► If DBD field definition is TYPE ‘X’

  Fill with x’00’
- ► If DBD field definition is TYPE ‘P’

  Fill low-order byte with x’0C’ and other bytes with x’00’
- ► If DBD field definition is TYPE ‘C’

  Fill with x’40’

When space without field definitions is added to a segment, use Fill with x’00’.

4.1.7 HALDB Alter operational considerations

To use HALDB Alter function in a data sharing environment, all IMS data sharing systems must be running IMS 13 and MINVERS must be set to value of "13.1" in DBRC. Type-2 command is used to start ALTER, which requires the Common Service Layer (CSL) to bet set up with a minimum of following components:

- SCI
- OM

Type-1 **/INITIATE OLREORG** command is not supported for ALTER. OLR processing is done for all partitions in a HALDB database when ALTER is requested. We recommend that you use member OLC to bring specific ACBLIBs online and combine HALDB ALTER with new DB versioning. For more information about DB versioning, see Chapter 6, "Database versioning" on page 181.

Other operational considerations include the enhanced **QUERY OLREORG** command, new OLR return, reason codes, and completion codes for ALTER, new "SF" status code, and the following new DFS and DSP messages:

- DFS1849E
- DFS3197I
- DFS3198I
- DFS3436E
- DFS3547E
- DSP0174E
- DSP0175E
- DSP1097E

The following DFS messages include changes for Alter:

- DFS047A
- DFS2991E

Also, there are several new **DBRC** commands and modified DBRC LIST command outputs for HALDB Alter.

HALDB Alter type-2 QUERY command

The **QUERY OLREORG** command is used to query if a HALDB is undergoing a structural change process. To query information about Alter processing, you can specify one or more partition names or the name of a HALDB master.

The following keyword parameters are changed or added to the type-2 **QUERY OLREORG** command:

- NAME()

  This keyword specifies the name of a HALDB master or the names of the HALDB PHDAM or PHIDAM partitions to be queried. NAME() is optional. A parameter with the wildcard character (*) is not allowed, except as NAME(*) for all defined HALDB partitions. NAME(*) is the default.

- STATUS()

  By using this keyword, you can display the online reorganizations that include the specified status.

- ALTER

  Displays status of alter processing for all partitions in a HALDB database that is being altered online.

- ALTDONE

  Displays all partitions for which alter processing is complete.

- ALTINPRG

  Displays all partitions that are being altered.

HALDB Alter return, reason, and completion codes

The new return, reason, and completion codes for the **INIT OLREORG OPTION(ALTER)** command are shown in Figure 4-8.

INITIATE OLREORG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	Command completed successfully.
X'00000010'	X'00004520'	Another OLR Alter in progress.

INITIATE OLREORG Completion Codes

Completion Code	Completion Code Text	Meaning
0		Command completed successfully
1E1	OLR ITASK creation failed	OLR internal ITASK can not be created
1E2	Incorrect HALDB version detected	The ddir version for the database is different than the number recorded in the RECON DB record for the HALDB
1E3	Partition queued for OLR	The partition is being queued for HALDB alter processing
1E4	HALDB alter pending for Online Change	An alter request against the same HALDB was done, but an Online Change has not yet done for that HALDB
1E5	No DB structure change detected	An alter request is made but there are no database structure changes made
1E6	Insufficient CI/Block size detected	During alter processing, the CI/Block size of the database data set is smaller than the database largest segment size
1E7	Unsupported DBD changes detected	An alter request is made but the type of DBD changes made are not supported
(many more)	...	...
1EC	Logical database error	During alter processing, an error related to logical relationship on the altered DBD is detected

Figure 4-8 New return, reason, and completion codes for the HALDB ALTER command

HALDB Alter new “SF” status code

IMS returns “SF” status code and continues to run normally when an application program that has field-level sensitivity attempted to read a database segment that was altered in length by the ALTER option of the HALDB online reorganization function.

Until the ACB members for the altered database are activated by the online change function, the altered segment cannot be accessed by application programs that have field-level sensitivity. You can use the member online change function to activate the ACB members for the altered database and rerun the application program

HALDB Alter DBRC LIST command output

In the DBRC LIST command output for a HALDB master database, the following information is stored in the DB and Partition RECON records about HALDB ALTER processing:

- ALTER COUNT indicates number of partitions that are to be altered.
- ALTER COMPLETE COUNT indicates the number of partitions for which the alter process completed.

Example 4-3 on page 115 shows an example of these fields as the last two lines of the output.

Example 4-3 LIST.DB output

```
DB
   DBD=DBOHIDK5                        DMB#=3      CHANGE#=3     TYPE=HALDB
   SHARE LEVEL=3             GSGNAME=**NULL**
   DBRCVGRP=**NULL**
   PSNAME=**NULL**  DBORG=PHIDAM   DSORG=OSAM    CURRENT PARTITION ID=00001
   FLAGS:                             COUNTERS:
     RECOVERABLE          =YES          PARTITIONS              =4
     ONLINE REORG CAPABLE  =YES         DATA SET GROUP MEMBERS  =2
                                        ALTER COUNT             =4
                                        ALTER COMPLETE COUNT    =2
```

LIST output for a HALDB database partition contains the following new information:

- ALTER BLOCK SIZE lists new block sizes to be used by the alter process.
- ALTER IN PROGRESS indicates whether the alter process started.
- PARTITION ALTERED indicates whether the alter process completed.

Example 4-4 shows the new information for HALDB partition with the new fields highlighted. ALTER BLOCK SIZE is shown only for OSAM databases.

Example 4-4 LIST.DB output for partition record

```
DB
   DBD=POHIDKA   MASTER DB=DBOHIDK5  IRLMID=*NULL  CHANGE#=3     TYPE=PART
   USID=0000000002  AUTHORIZED USID=0000000002  HARD USID=0000000002
   RECEIVE USID=0000000002  RECEIVE NEEDED USID=0000000000
   DSN PREFIX=IMSTESTS.DBOHIDK5                         PARTITION ID=00001
   PREVIOUS PARTITION=**NULL**    NEXT PARTITION=**NULL**
   OLRIMSID=**NULL**    ACTIVE DBDS=M-V
   REORG#=00000
   ONLINE REORG STATISTICS:
       OLR BYTES MOVED = 5576000
       OLR SEGMENTS MOVED = 16000
       OLR ROOT SEGMENTS MOVED = 4000

   FREE SPACE:
     FREE BLOCK FREQ FACTOR=0    FREE SPACE PERCENTAGE=50

   PARTITION HIGH KEY/STRING (CHAR):              (LENGTH=5  )
     .....
   PARTITION HIGH KEY/STRING (HEX):
     FFFFFFFFFF

   OSAM BLOCK SIZE:
     A = 4096
     B = 4096
   ALTER BLOCK SIZE:
     A = 0
     B = 8192

   FLAGS:                             COUNTERS:
     BACKOUT NEEDED       =OFF          RECOVERY NEEDED COUNT   =0
     READ ONLY            =OFF          IMAGE COPY NEEDED COUNT =0
     PROHIBIT AUTHORIZATION=OFF         AUTHORIZED SUBSYSTEMS   =1
                                        HELD AUTHORIZATION STATE=3
```

```
                                        EEQE COUNT                =0
     TRACKING SUSPENDED    =NO          RECEIVE REQUIRED COUNT    =0
     OFR REQUIRED          =NO          OLR ACTIVE HARD COUNT     =0
     PARTITION INIT NEEDED =NO          OLR INACTIVE HARD COUNT   =0
     OLREORG CURSOR ACTIVE =NO
     PARTITION DISABLED    =NO
     ONLINE REORG CAPABLE  =YES
     REORG INTENT          =NO
     QUIESCE IN PROGRESS   =NO
     QUIESCE HELD          =NO
     ALTER IN PROGRESS     =NO
     PARTITION ALTERED     =NO
```

LIST output for a REORG also contains new information. REORG record now indicates whether the HALDB partition DBDS was altered during the online reorganization. REORG record has the text ALTER if database is altered, as shown in Example 4-5.

Example 4-5 REORG record for a partition that was altered

```
REORG
    RUN     = 12.063 12:19:55.476258         *    USID = 0000000002
    REORG#  = 00005                                         ALTER
    STOP    = 12.063 12:20:13.210119           ONLINE   RECOV = NO
```

HALDB Alter DBRC API query output

Some of the DBRC API query output changed. The following new fields are returned when the OLR ALTER function is being used:

- DSPAPQHP - HALDB Partition output block:
  - Now returns the array of new Block/CI sizes to be used by the OLR alter process
  - Exists only if the alter process is still in progress
  - Two new flags:
    - One indicates whether Alter is in progress
    - One indicates whether the alter is complete for the partition
- DSPAPQHB - HALDB output block two new counters:
    - Total number of partitions being altered
    - Number of partitions for which the alter process completed
- DSPAPQRR – DBDS reorganization output block

  A new flag that indicates whether an online reorganization altered the HALDB.

Note: Some of the DBRC records changed in support of the OLR ALTER process. If you have programs that access these records, reassemble them by using the following changed DBRC control blocks:

- DSPDBHRC – Database record (DB)

  No size increase
- DSPPTNRC – Partition record (PART)

  Size increases only when ALTER is running
- DSPRRGRC – Reorganization record (REORG)

  No size increase

HALDB Alter DBRC command changes

The following DBRC commands are used internally by IMS to set flags for ALTER processing. IMS users should not have to use these commands (unless required) to clean up RECON records after a major system failure:

- **`CHANGE.DB`**

  ALTER | NOALTER:

  – New, optional keywords
  – Specifies whether the HALDB partition is being altered
  – Cannot be changed if the partition is authorized

- **`NOTIFY.REORG`**

  ALTER:

  – New, optional keyword
  – Specifies that OLR altered the database structure
  – Indicates that OLR is altering the database structure
  – ALTER can be specified only with ONLINE

HALDB Alter security considerations

Because the new ALTER form of Online Reorg can change a database structure, we recommend that the **`INITIATE OLREORG`** command is secured. Table 4-1 shows the RACF attributes that you can use for securing the type-2 **`INIT`** command.

Table 4-1 RACF attributes for INIT command

IMS command	Keyword	RACF access authority	Resource name
INIT	OLREORG	UPDATE	IMS.plxname.INIT.OLREORG

HALDB Alter performance considerations

When a HALDB is reorganized with the ALTER option, the performance should be same as reorganizing a HALDB without the ALTER option.

Overall elapsed time to complete an **`INIT OLREORG OPTION(ALTER)`** command for a HALDB database is likely to be greater than an **`INIT OLREORG`** for a single HALDB database partition, or even a subset of partitions. The ALTER option applies to ALL partitions in a HALDB database in one single pass.

If BLKSIZE for a PHDAM database changes, IMS takes an extra lock per RAP during ALTER processing.

4.2 DEDB Alter

In IMS 13, Fast Path added the ability to dynamically change certain DEDB specifications. For example, the Units of Work (UOW), SIZE, ROOT, and Randomizer routine can be changed while the DEDB is online with the new DEDB Alter utility. This support is available for VSO areas if the areas are unloaded first by using the **/VUNLOAD** command. Allowing dynamic changes to the DEDBs improves data availability and reduces system downtime.

The DEDB Alter utility supports the following new functions:

- ALTERAREA is used to change the UOW, ROOT, SIZE, and RMNAME (randomizer name) values.
- REPLRAND is used to specifically change the Randomizer name.

The randomizer must be a two-stage randomizer where the logic in the randomizer behaves like a two-stage randomizer. If the number of RAPs change in the AREA being altered, it must not affect the randomization within the DEDB (the AREA selected) or within all AREAs not being altered. Also, the DEDB Area must be registered to DBRC. This configuration supports ACBSHR=Y for the sharing IMS system if the active system is XRF or FDBR.

For the DEDB Alter functions, all IMS data sharing systems must be at the IMS 13 level. There are no special hardware requirements for DEDB Alter. Finally, the DBRC MINVERS parameter must be set to 13.1 to use the DEDB Alter capability.

4.2.1 Preparation for DEDB Alter

There is some preparation that is needed before the DEDB Alter utility can make the DEDB changes. For the ALTERAREA function, the active DEDB AREA statement must be modified with the new SIZE, UOW, or ROOT definitions. If the RMNAME parameter is used, the new randomizer name must be different from the existing randomizer name. If the randomizer is being changed (with REPLRAND or ALTERAREA), the new randomizer must be assembled and the link edited into the IMS SDFSRESL STEPLIB concatenation. The active DEDB AREA statement must be modified. The active randomizer must be a two-stage randomizer and the new active randomizer must also be a two-stage DEDB randomizer. The new randomizer also must function in the same way as the old randomizer. That is, for a specific key, both randomizers must randomize to the same area and RAP (root anchor point).

After the DEDB AREA statements are modified, the DBDGEN utility is run to create DEDB DBD definitions. The ACBGEN utility is run next for all PSBs that reference the changed DEDB DBD. The new ACBs that result from the ACBGEN are added to the staging ACBLIB data sets.

The ACBLIB staging library must have a dynamic allocation member. A DFSMDA member can be created for the ACBLIB staging library if one does not exist by using the DFSMDA macros. An example of DFSMDA macros for the staging ACBLIB is shown in Example 4-6.

Example 4-6 DFSMDA macros for the staging ACB library

```
DFSMDA TYPE=INITIAL
DFSMDA TYPE=IMSACB,DSNAME=IMS.STAGING.ACBLIB
DFSMDA TYPE=FINAL
```

There is a new parameter in the <SECTION=FASTPATH> area of the DFSDFxxx proclib member that is called ALTERGRP=nnnnn. By using this parameter, the user can define a new data sharing group name, which allows data sharing partners to communicate with each other during DEDB Alter utility execution. ALTERGRP= contains 1 - 5 alphanumeric characters that are prefixed by DBF. The DEDB alter data sharing group that data sharing IMS systems join is DBFxxxxx. There is no default for this parameter.

During the DEDB ALTER run, DEDB Alter commits the new DBD and moves it into the active ACBLIB. In a data sharing environment, the IMS system can share the ACBLIB with the other IMS subsystems in the data sharing environment or each IMS can have its own ACBLIB. The ACBSHR=Y|N parameter indicates how the ACBLIB is used between the IMS subsystems. The ACBSHR specification is found in the FASTPATH section in the DFSDFxxx proclib member if the Common Service Layer (CSL) is used in the environment. If CSL is not used, the ACBSHR specification is found by using the following precedence:

- The first check is in the DFSCGxx proclib member.
- The second check is in the COMMON_SERVICE_LAYER section in the DFSDFxxx proclib member.
- The third check is in the FASTPATH section in the DFSDFxxx proclib member.

Although DEDB Alter uses the ACBSHR setting that is found in the local IMS subsystem, all IMS subsystems in the data sharing environment must have the same ACBSHR parameter setting.

The next step in the preparation for DEDB Alter is to allocate both the Shadow Area data sets and the Shadow Image Copy data sets. The Shadow Area data sets are used for migrating the existing data from the Active Areas to the Shadow Areas. The Shadow Area data set can be a Single Area Data Set (SADS) or it can be a Multiple Area Data Set (MADS) if there are two to six Shadow Area data sets allocated. A SADS also can be turned into a MADS after the DEDB Alter is run by allocating more Shadow Area data sets. The Shadow Area data sets are used exclusively by the DEDB Alter utility and are not accessible by the IMS subsystem until DEDB ALTER completes and swaps in new ADSs. The Shadow Area Image Copy data sets are created during the migration of data to the Shadow Area data sets.

Before the DEDB Alter utility is run, the Shadow Area and Shadow Image Copy data sets must be allocated. After they are allocated, they must be registered to DBRC before DEDB ALTER is run by using the `INIT.ADS` command. The DEDB Area Initialization utility (DBFUMIN0) formats the Shadow Area and Shadow Area Image Copy data sets and flags them as available in the RECON data set. There must be at least one Shadow Area and Shadow Area Image Copy data sets flagged as Available in the Recon.

The DEDB Area Initialization Utility (DBFUMIN0) was enhanced to format the Shadow Area data sets and the Shadow Area Image Copy data sets. The following two new control cards for the DBFUMIN0 utility are available:

- ACTIVE

  The ACTIVE keyword indicates that the DEDB Area data sets are to be formatted. This is the default.

- SHADOW

  The SHADOW keyword indicates that the Shadow Area and Shadow Area Image Copy data sets are to be formatted.

This utility formats the Active DEDB Area data sets or the Shadow Area and Shadow Area Image Copy data sets in one execution, but it cannot perform both types of data sets in one execution. If DBRC=Y, it can format the Shadow Area data sets and the Shadow Area Image Copy data sets in one execution. The Shadow Area and Shadow Area Image Copy data sets can be formatted while the Active DEDB Area data sets are online. After the utility completes, the following flags are set in the DBRC Recon data set:

- The Shadow Area data sets are flagged as SHADOW AVAIL
- The Shadow Area Image Copy data sets are flagged as SHADOW IC AVAIL

The Shadow Area and Shadow Area Image Copy data sets must be formatted before the DEDB Alter utility can run.

4.2.2 DEDB Alter Utility control statements

The DEDB Alter utility uses the following control cards for execution:

- TYPE ALTER

  The TYPE ALTER starts the DEDB Alter utility.

- ALTERAREA or REPLRAND

  ALTERAREA and REPLRAND are the two functions.

- UNKEYSEG

  The UNKEYSEG keyword determines whether unkeyed segments are allowed in the DEDB.

  There are three options on the UNKEYSEG parameter: ALL, NONE, and ISRTFILA. One option must be specified.

- TIMEOUT

  The timeout value specifies the maximum time ALTER holds up online activity during serialization periods. When the function times out, online activity continues. If DEDB ALTER times out, the retry and retrywait options determine if ALTER continues or ends.

- RETRY NO/YES/retry_value (1-99)

  The RETRY keyword indicates whether to retry the DEDB Alter function if a timeout occurs:

  - NO indicates that there should be no retries after the timeout occurs.
  - YES indicates that the retries should continue until the utility is successful.
  - retry_value indicates the number of retries that can be attempted after the timeout occurs.

- RETRYWAIT

  The RETRYWAIT keyword indicates the number of seconds to wait before the next retry of the commit process if RETRY NO is not specified.

- GO

  The GO keyword starts the DEDB Alter utility execution.

4.2.3 DEDB Alter Utility execution (ALTERAREA)

Example 4-7 shows the sample JCL with the DEDB Alter execution control cards for the ALTERAREA function.

Example 4-7 Sample JCL for the DEDB Alter

```
//ALTAREA  JOB ….
//FPUTIL   PROC  SOUT=A,RGN=1M,
//               DBD=,REST=00,DIRCA=002,
//               PRLD=,IMSID=,AGN=,SSM=,ALTID=
//FPU      EXEC  PGM=DFSRRC00,REGION=&RGN,
//               PARM=(IFP,&DBD,DBF#FPU0,&REST,00,,1,
//               &DIRCA,&PRLD,0,,,,&IMSID,&AGN,&SSM,,
//               &ALTID)
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT,…
//S0          EXEC FPUTIL,RGN=1M,DBD=DEDBJN21,REST=00,IMSID=IMS1
//SYSIN       DD   *
TYPE          ALTER
ALTERAREA     DB21AR0
RETRY         NO
TIMEOUT       30
GO
/*
```

The DEDB Alter utility ALTERAREA function changes the UOW, SIZE, ROOT values while the DEDB Area data set is online. The SIZE parameter affects the CI size of an Area. The UOW and ROOT parameters affect the Root Addressable and Independent Overflow parts of the DEDB Area.

The RMNAME parameter changes the Randomizer name while the active DEDB Area data set is online. When a Randomizer name is being changed, the Active DEDB Area data sets are read by using the existing randomizer routine. The data is migrated from the Active DEDB Area data sets to the Target DEDB Area data sets by using the new Randomizer name.

After the DEDB Alter is completed, the new randomizer name replaces the existing randomizer name. Then, all DEDB Areas start to use the new randomizer name. The new name must be a two-stage randomizer and it must be a different name from the existing randomizer name. However, both randomizers must randomize to the same area and root anchor point (RAP) to avoid inaccessible data in the other areas of this DEDB.

There can be only one Active DEDB Area data set changed at a time. It is not possible to run the DEDB Alter utility concurrently for another DEDB Area in the same DEDB database.

The ALTERAREA function does not support changes to DEDB databases with SDEPs. However, it is possible to replace the randomizer name by using the REPLRAND function for DEDBs with SDEPs.

Relationship of UOW, ROOT, and SIZE Parameters

The following UOW, ROOT, and SIZE parameters can be changed by the DEDB Alter utility:

- UOW=(number1,overflow1):
  - Number1 = number of Control Intervals (CI) in a UOW
  - Overflow1 = number of Control Intervals in Dependent Overflow section of UOW
- ROOT=(number2,overflow2)

  Space that is allocated to Root Addressable Part and Independent Overflow:
  - Number2 = Space (in UOWs) for Root Addressable Part and Independent Overflow
  - Overflow2 = Space (in UOWs) for Independent Overflow
- SIZE=value

  The SIZE parameter indicates the size of the CI in bytes:
  - 512 bytes, 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, 2 0KB, 24 KB, and 28 KB
  - SIZE value must match the CI size that is defined to VSAM

The *direct part* of a DEDB area consists of the root addressable part and the independent overflow part. These parts can contain both root segments and their direct dependents, but not sequential dependent segments. The root addressable part is further divided into groups of CIs that are called UOWs. The independent overflow part holds roots and DDEPs that overflowed from the direct part, which is similar to (but not quite the same as) the way overflow works for HDAM. If the RAP is full, the data first goes to the dependent overflow in the same UOW. If dependent overflow is full, it goes to independent overflow.

Following the direct part is a group of CIs that are called the reorganization UOW. These CIs are no longer used by Fast Path, but are still there for compatibility reasons.

Whatever space is left over from the direct part and reorganization UOW is used for SDEPs. There are no parameters in the DBD to define how much space is to be used for SDEPs; it is only the difference between the VSAM DEFINE and what is used by the other parts.

Figure 4-9 on page 123 shows the structure of DEDB Area.

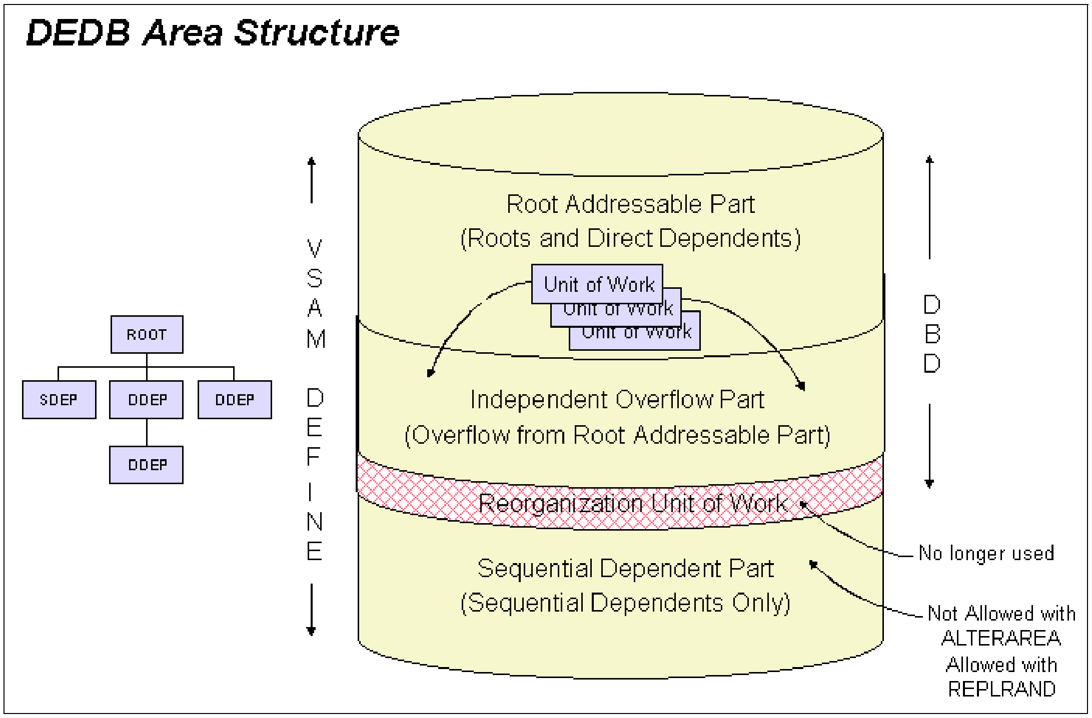


Figure 4-9 DEDB Area structure

Figure 4-10 shows some terminology that applies to a DEDB area.

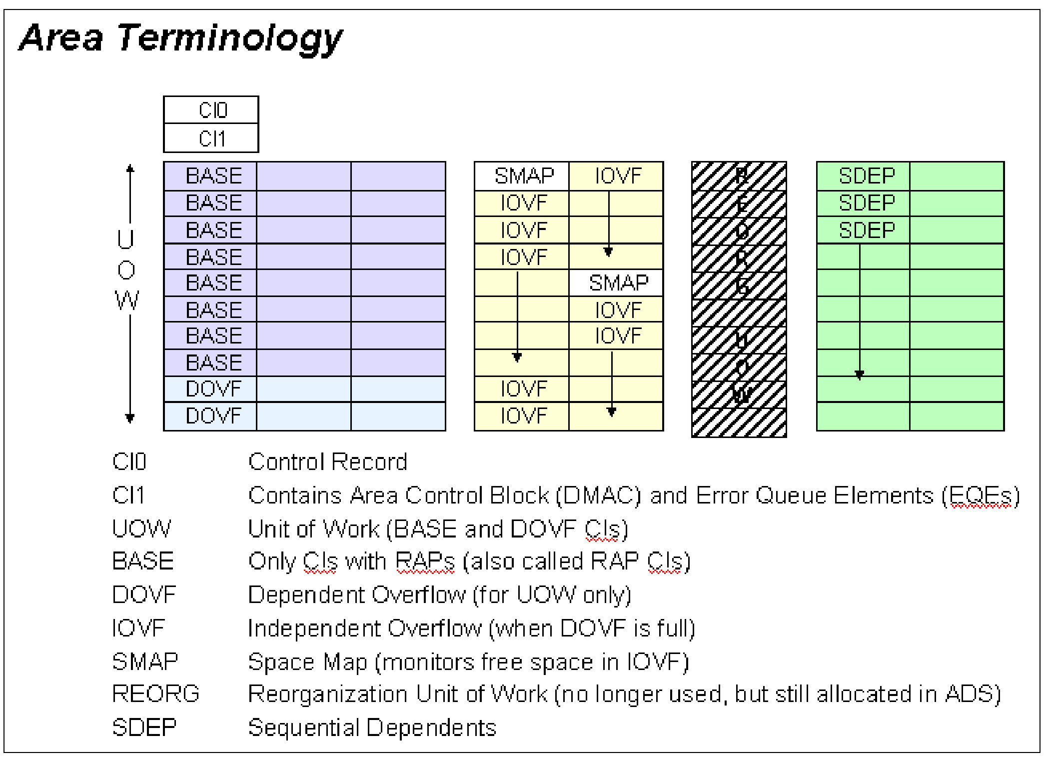


Figure 4-10 DEDB Area terminology

The first two CIs (CI0 and CI1) in an ADS contain control information and a control block that is called the DMAC.

The UOW that is shown in Figure 4-9 on page 123 consists of two types of CIs - BASE (sometimes called RAP) CIs and DOVF (dependent overflow) CIs. The DOVF CIs are used only for the overflow of BASE CIs in the same UOW.

Independent overflow also contains two types of CIs: Space Map (SMAP) CIs, which have a similar function to the bitmaps in HDAM or HIDAM databases, and the IOVF CIs, which contain roots and DDEPs that overflowed from the UOWs and their DOVF CIs,

The REORG UOW is no longer used by Fast Path, but exists in the data set for compatibility reasons.

SDEP CIs are all of the CIs from the end of the REORG UOW to the end of the VSAM ESDS.

Examples of increasing DEDB Area size

How the size of the DEDB Area can be changed is shown in the following examples:

- Expand the DEDB Area by using same Control Interval (CI) size

  DEDB Area size is increased without changing the CI size. Instead, the UOW parameter is changed to increase the number of CIs in a UOW. The ROOT parameter is also changed to allocate more space for the Root Addressable Part and the Independent Overflow.

- Expand the DEDB Area by using different Control Interval (CI) size

  DEDB Area size is increased y using a different CI size. In this example, the SIZE parameter uses a larger CI size, the UOW is increased to put more CIs in the UOW, and the ROOT parameter is increased to allocate more space for the Root Addressable Part and the Independent Overflow. If you increase the CI size, you do not have to change the ROOT and UOW parameters. You have the same number of UOWs and CIs, but the CIs is larger. Therefore, if you had data in overflow, it can end up in the RAP it randomizes to because it is no longer full.

Each area in a DEDB is defined by an AREA statement, which replaces the DATASET statement that is used for HDAM and HIDAM databases. The AREA statement defines the area name (or DD name if the area is not registered), the CI size, the size and configuration of a UOW, and how much independent overflow to allocate. The values of these parameters can be different for each AREA.

The AREA statements, SEGM statements, and FIELD statements define the hierarchical structure of the database. All areas have the same hierarchical structure. An example of DEDB Alter area is shown in Figure 4-11 on page 125.

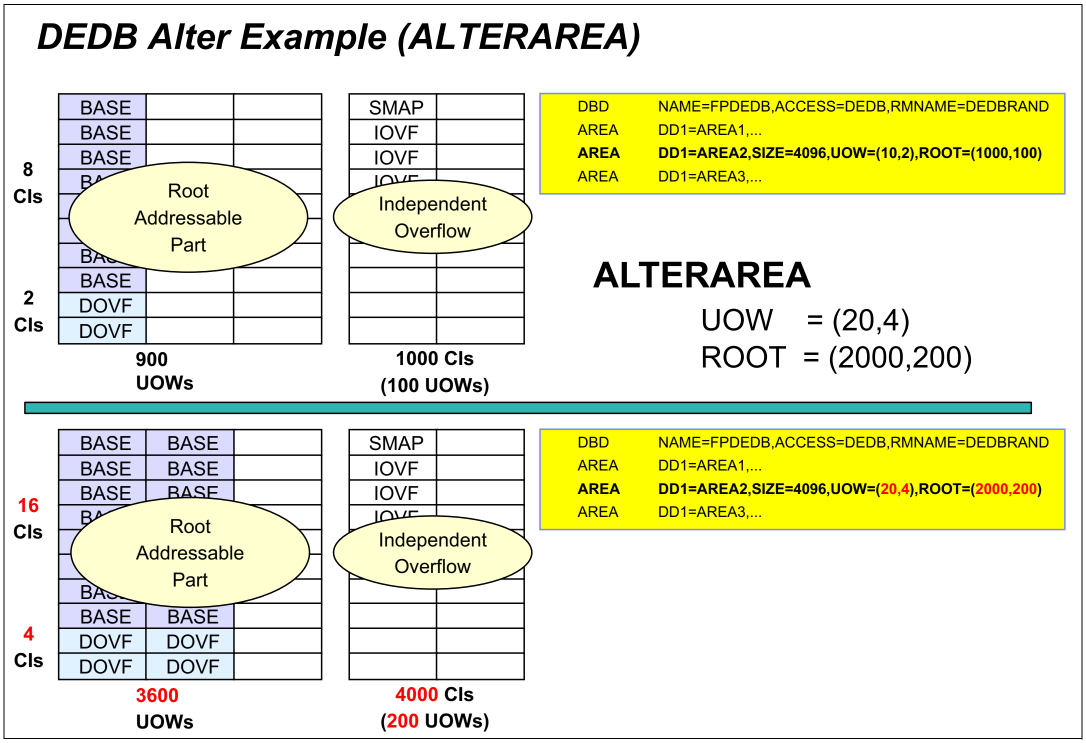


Figure 4-11 DEDB Alter example

After the DEDB DBD is altered, the DBDGEN and the ACBGEN are run, the Shadow Area and Shadow Area Image Copy data sets are allocated and registered to DBRC and formatted, it is time to run the DEDB Alter utility. The ALTERAREA function migrates the data from the Active DEDB Area data sets to the Shadow Area data sets and creates the Shadow Area Image Copy data sets. In the process, any changes to the UOW, ROOT, or SIZE parameters in the Staging ACBLIB are implemented. The current randomizer is used to read the Active DEDB Areas, but the new randomizer is used to insert to the Shadow Area data sets.

When the changes are committed, the Active DEDB Area data sets are quiesced and any DL/I calls are suspended after the data is migrated to the shadow ADSs. The Shadow Area data sets are synchronized with the Active DEDB Area data sets. The changed ACB in the Staging ACBLIB is moved to the Active ACBLIB and, if there is a new randomizer, it replaces the existing randomizer. The Shadow Area data set becomes the new DEDB Area data set preserving the original DEDB Area data set. Finally, the DEDB Area is unquiesced resuming suspended DL/I calls.

If the DEDB Alter function is successful, the Shadow Area data set is promoted to the Active Area data set replacing the previous Active Area data sets, which become the Shadow Area data sets. The Shadow IC data set is promoted to a User Image Copy. In DBRC, this User Image Copy is registered as a User Image Copy so that it can be counted in the GENMAX count.

The User Image Copy is not a Standard Image copy. It is an image of the active area data set. To recover a DEDB Area with this User Image Copy, the user must notify DBRC (NOTIFY.RECOV) with the name of this Area data set and the time when the Shadow Image Copy was created (that is, RCVTIME). Specifying the `GENJCL.RECOV` command without an image copy name picks up this Shadow Image Copy name.

If the DEDB Alter was unsuccessful, the Active Area data sets remain active and are still accessible to the IMS systems. The flags in DBRC are checked to see whether any data was written to the Shadow Area data sets. If the flag still shows AVAIL, no data was written to the Shadow Area data sets and they can be used in subsequent DEDB Alter utility executions. If the flag shows UNAVAIL, data was written to the Shadow Area data sets and they must be reallocated and reformatted.

4.2.4 DEDB Alter Utility Execution (REPLRAND)

A sample JCL that shows the DEDB Alter execution control cards for the REPLRAND function is shown in Example 4-8.

Example 4-8 A sample JCL for the DEDB Alter REPLRAND function

```
//ALTAREA  JOB ….
//FPUTIL   PROC  SOUT=A,RGN=1M,
//               DBD=,REST=00,DIRCA=002,
//               PRLD=,IMSID=,AGN=,SSM=,ALTID=
//FPU      EXEC  PGM=DFSRRC00,REGION=&RGN,
//               PARM=(IFP,&DBD,DBF#FPU0,&REST,00,,1,
//               &DIRCA,&PRLD,0,,,,&IMSID,&AGN,&SSM,,
//               &ALTID)
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT,…
//S0       EXEC FPUTIL,RGN=1M,DBD=DEDBJN21,REST=00,IMSID=IMS1
//SYSIN    DD   *
TYPE       ALTER
REPLRAND
RETRY      NO
TIMEOUT    30
GO
/*
```

By using the DEDB Alter function REPLRAND, the user can change the randomizer name by using the RMNAME DEDB DBD while the DEDB remains online. The new name and the existing name must be a two-stage randomizer and the new name must be different from the existing randomizer name.

After the DEDB Alter completes, the new randomizer replaces the existing randomizer and all Active Areas in the DEDB database begin to use the new randomizer. The new randomizer also must function in the same way as the old randomizer; that is, both randomizers must randomize to the same area and RAP. There can be only one Active DEDB changing at a time. The REPLRAND function supports DEDB databases with and without SDEPs.

The DEDB Alter utility with the REPLRAND function runs after the new randomizer is assembled and linkedited, the DEDB DBD is modified with the new randomizer name, and the DBDGEN and ACBGEN are run. The DEDB Alter REPLRAND function loads the new randomizer from the IMS SDFSRESL STEPLIB concatenation and the ACB from the Staging ACBLIB. When the randomizer change is committed, the Active DEDB database is quiesced, which forces DL/I calls to be suspended and allows the changed ACB in the Staging ACBLIB to be moved to the Active ACBLIB. Finally, the DEDB database is unquiesced and DL/I calls are resumed.

If the DEDB Alter REPLRAND function is successful, the new randomizer replaces the existing randomizer. If the function is unsuccessful, the existing randomizer remains in effect.

4.3 Fast Path secondary index enhancement

In IMS 12, IMS Fast Path added the ability to create secondary indexes. In IMS 13, this function is enhanced to allow segment search arguments to use the Boolean operators AND and OR. Also, support was added to allow specific command codes to be used with the Secondary Index search field.

Better programming capabilities for DEDBs are possible because of this support. DL/I calls also can be refined and more command code support is now available.

4.3.1 Example of Fast Path secondary indexes

The DBD for the DEDB (DEDBHOSP) is shown in Figure 4-12. Segment HOSPITAL is the source and the target segment. The LCHILD segment specifies the secondary index segment, IXSASEG, and database, FPSI1ASA. PTR=SYMB is specified, as required. The XDFLD statement specifies the name of the search field, HOSPADDR, for use with the secondary index. It also specifies that field IXSAIDX is used to build the search field.

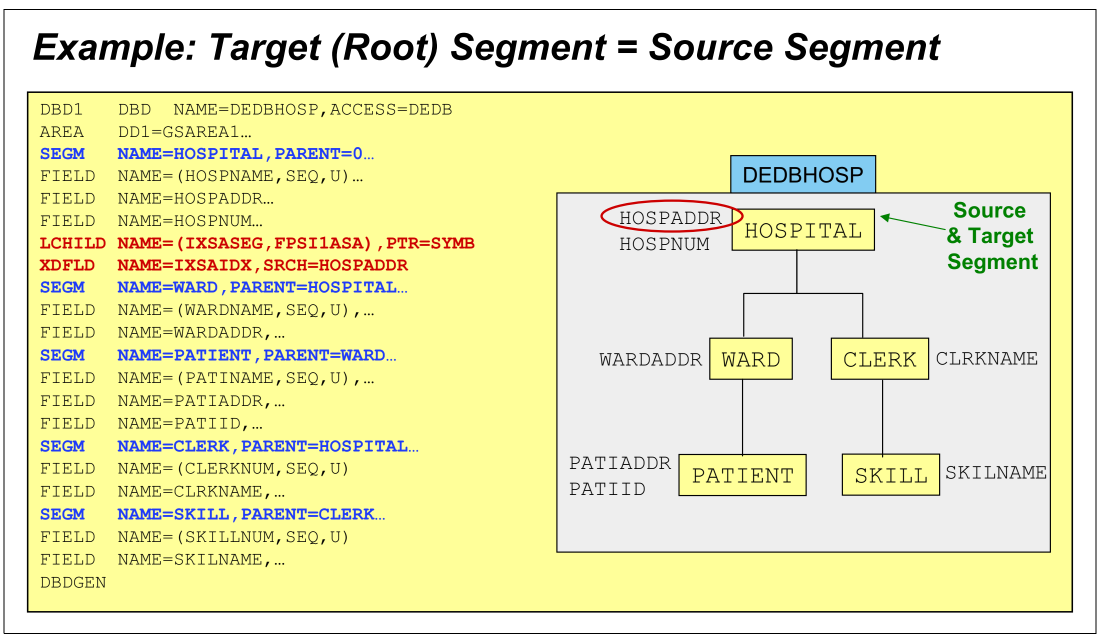


Figure 4-12 Hospital database with secondary index defined

The use of a secondary index means that the database hierarchical structure can be seen in different views. When the source segment and the target segment are the same segment, the view is still the physical view, although the secondary index key is used to retrieve the segment. Figure 4-13 shows an example where the view of the structure that uses the secondary index is the same as the physical view. The key feedback area that is received by the program is composed of the secondary index key and the keys of the dependent segments. The feedback area for the PATIENT segment is the secondary index key and the keys of the WARD and PATIENT segments.

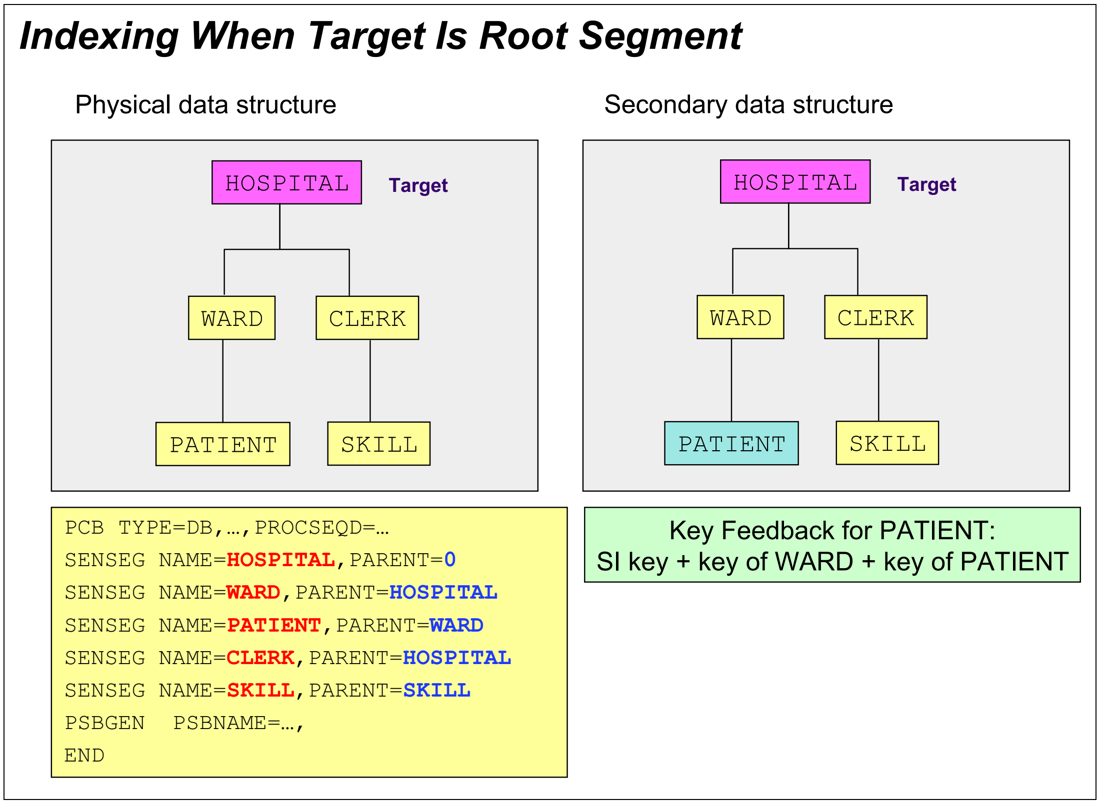


Figure 4-13 Secondary index with Source and Target segments the same

The secondary index is shown in Figure 4-14. It uses the LCHILD statement to establish the relationship with the HOSPITAL database. The IXSAIDX field from the XDFLD field is named.

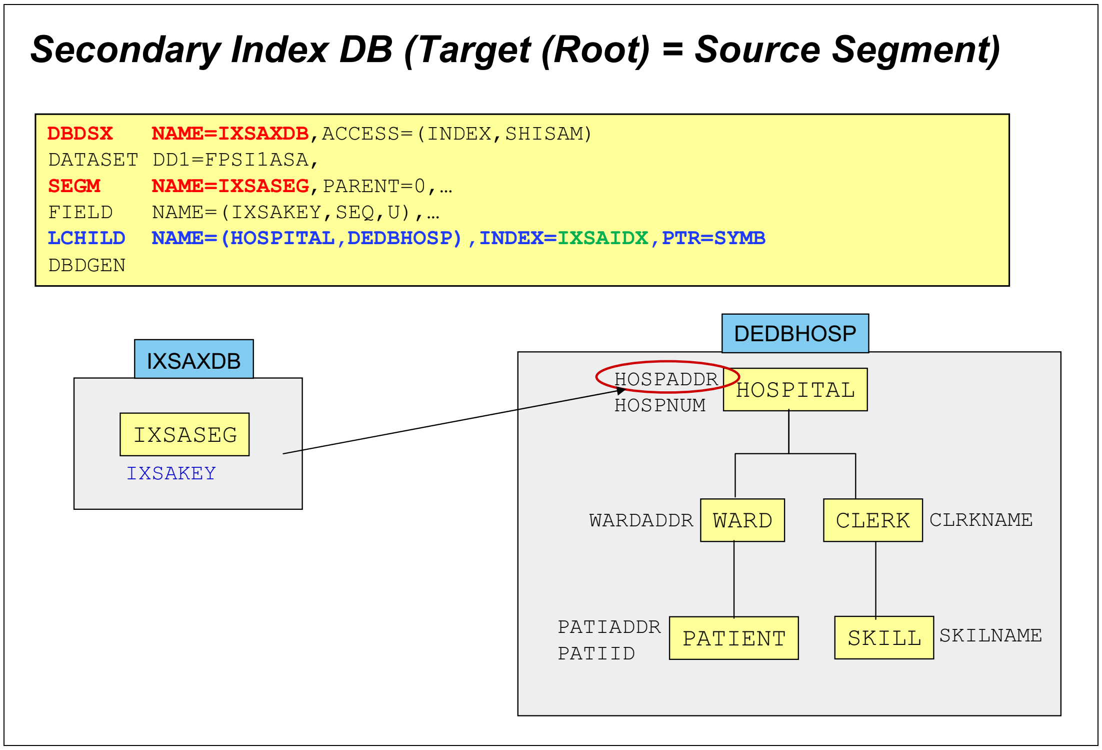


Figure 4-14 Secondary index with LCHILD statement to the HOSPITAL database

When the secondary index uses the source fields from one segment (called the *source segment*) but retrieves another segment (called the *target segment*), the XDFLD statement uses the SEGMENT parameter to indicate that the named fields are not on the segment that the XDFLD is under. The LCHILD and XDFLD statements are coded under the TARGET segment, as shown in Figure 4-15.

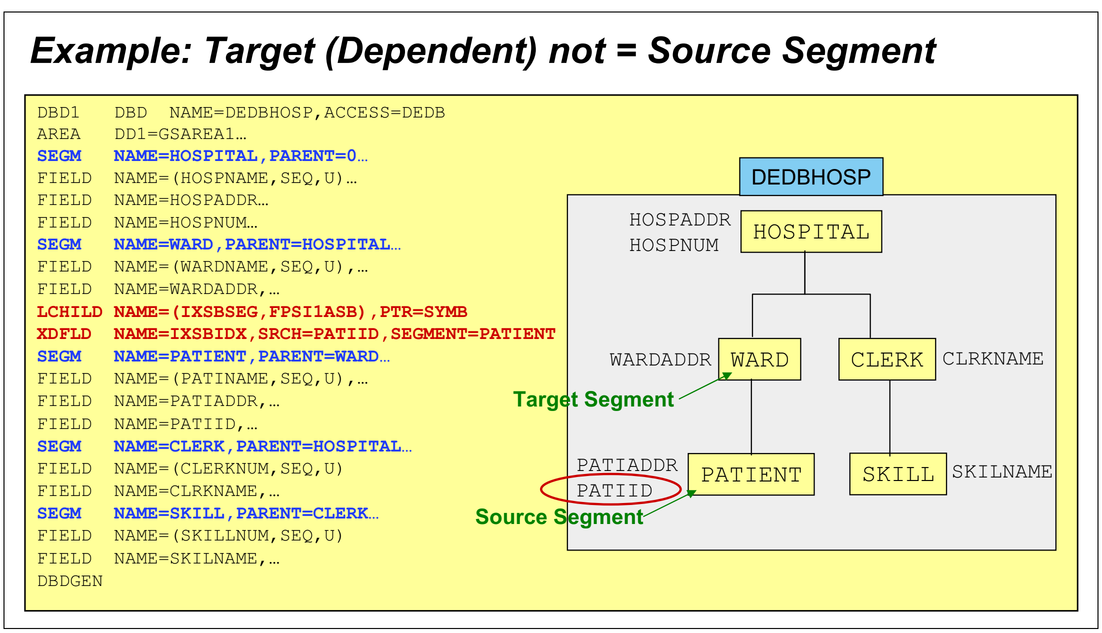


Figure 4-15 HOSPITAL database with secondary index with different SOURCE and TARGET

The secondary index view now is different from the physical view. The WARD segment that is the TARGET segment of the secondary index is now the root segment, as shown in Figure 4-16. The key feedback area that is received by the program also is now altered. The key feedback for the HOSPITAL segment is the secondary index key and the key of the HOSPITAL segment. The key feedback for the PATIENT segment is the secondary index key and the key of the PATIENT segment.

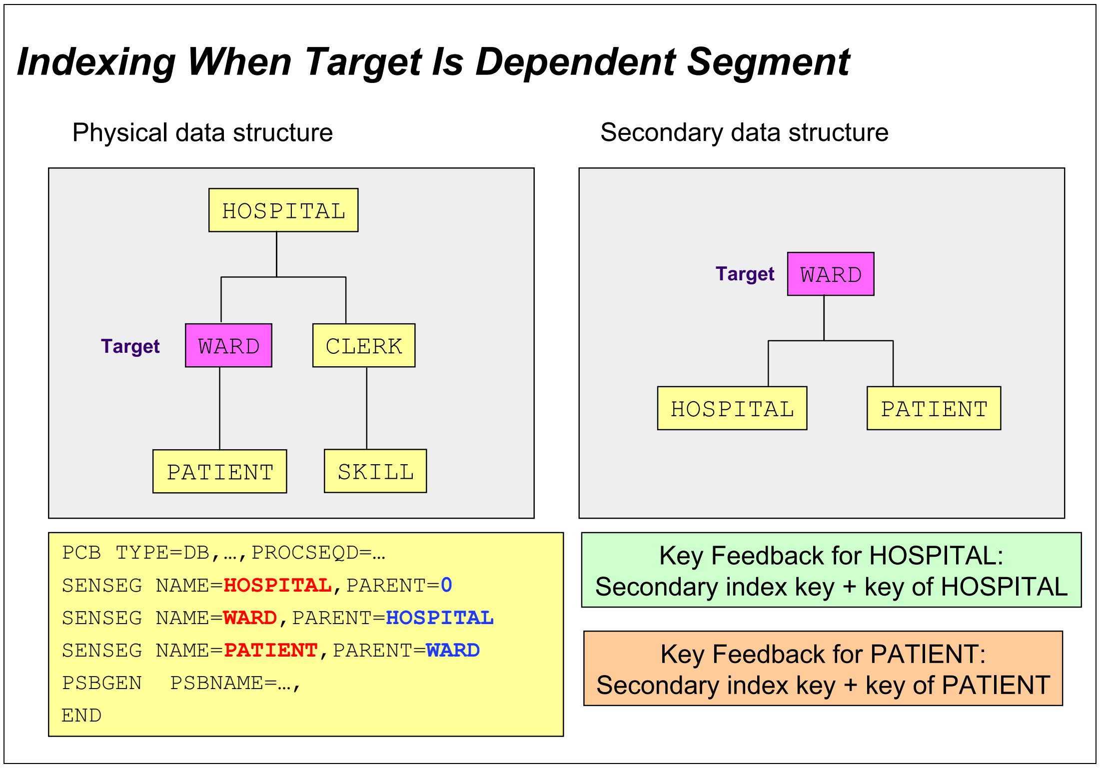


Figure 4-16 Secondary index view with TARGET segment

The secondary in index that is shown in Figure 4-17 shows the index for new view of the structure.

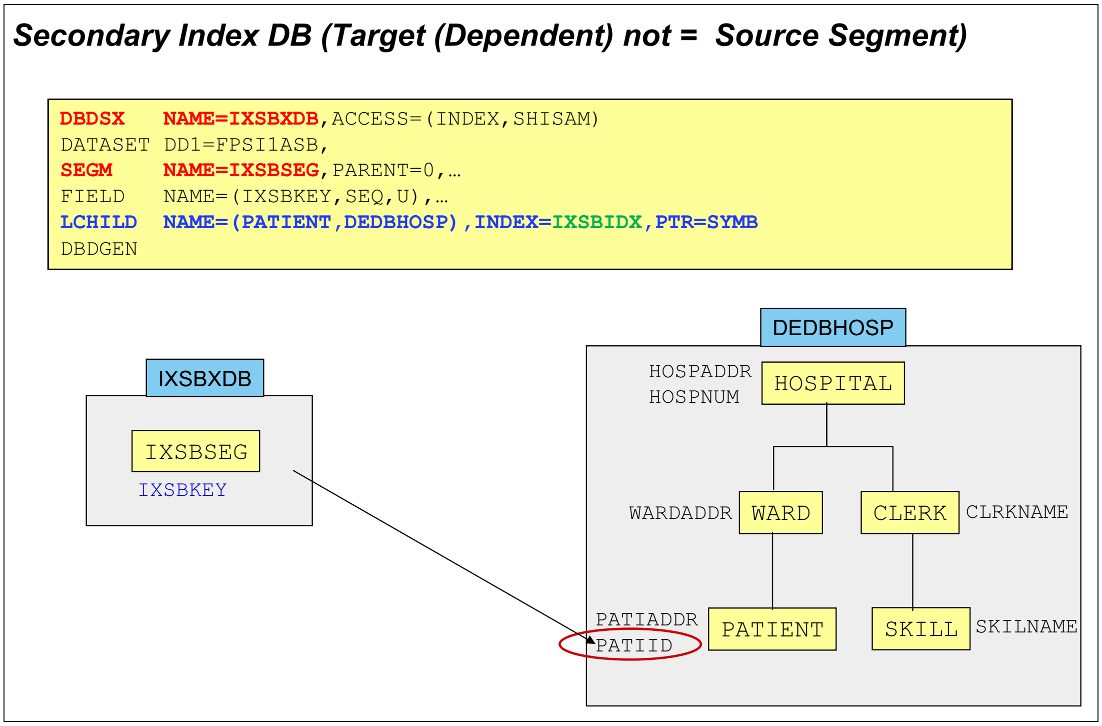


Figure 4-17 Secondary index segment for different SOURE and TARGET

4.3.2 Boolean Operators

IMS V13 now support the use of Boolean operators on DLI calls now, including the AND and OR operators. Then, AND is specified with a “ * ” or “&” symbol and the OR is specified with the “+” o r “|”, which allows more complex calls that use the secondary indexes. A search can be made looking for either values by using the OR clause, as shown in Figure 4-18.

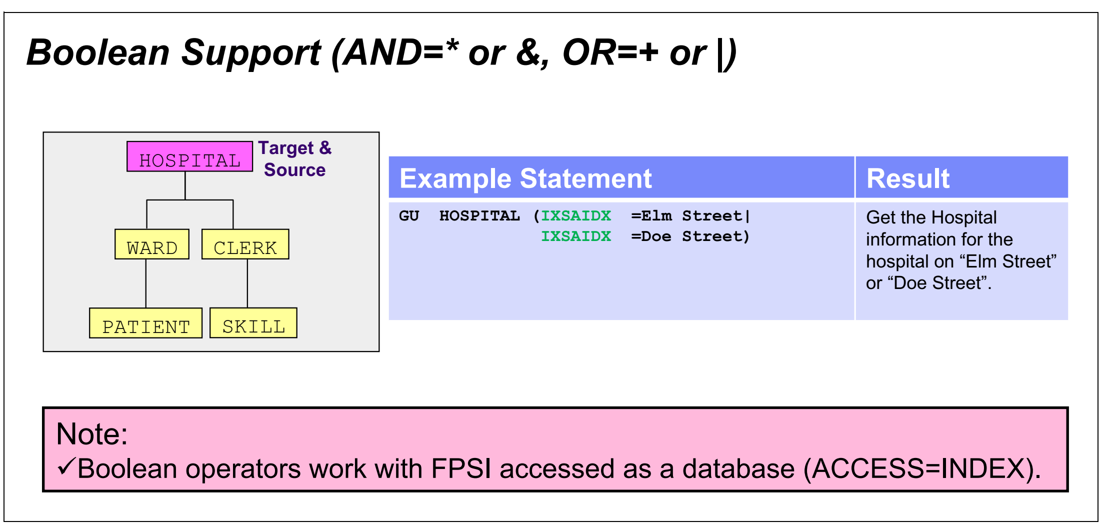


Figure 4-18 Secondary index call with Boolean operator OR

4.3.3 New command codes

There are several new command codes to support Fast Path secondary indexing, as shown in Figure 4-19. Some of the command codes also are restricted for use with secondary indexing.

Supported Command Codes

Command Code	Explanation
C	Use the concatenated key of a segment to identify the segment
D	Retrieve or insert a sequence of segments in a hierarchic path using only one call, instead of using a separate (path) call for each segment.
F	Back up to the first occurrence of a segment under its parent when searching for a particular segment occurrence. Disregarded for a root segment.
L	Retrieve the last occurrence of a segment under its parent.
N	Designate segments that you do not want replaced when replacing segments after a Get Hold call.
P	Set parentage at a higher level than what it usually is (the lowest-level SSA of the call).
Q	Reserve a segment so that other programs cannot update it until you have finished processing and updating it.
U	Limit the search for a segment to the dependents of the segment occurrence on which position is established.
V	Use the hierarchic level at the current position and higher as qualification for the segment.
-	Null. Use an SSA in command code format without specifying the command code.

Figure 4-19 Supported command codes

The unsupport3d command codes are shown in Figure 4-20.

Unsupported Command Codes

Command Code	Explanation	Status Code
A	Clear positioning and start the call at the beginning of the database	AJ
G	Prevent randomization and search the database sequentially	AJ
M	Move a subset pointer to the next segment occurrence after your current position	AJ
O	Qualify by Position	AD
R	Retrieve the first segment occurrence in a subset	AJ
S	Unconditionally set a subset pointer to the current position	AJ
W	Set a subset pointer to your current position, if the subset pointer is not already set	AJ
Z	Set a subset pointer to 0, so it can be reused	AJ

Figure 4-20 Unsupported command codes

4.4 DBRC DELETE.LOG INACTIVE and TOTIME commands

IMS 13 adds better control for the **DELETE.LOG** command. The **DELETE.LOG** command uses Stop Time for when INACTIVE and TOTIME parameters are used with the command. The **DELETE.LOG** command also determines when the PRILOG or SECLOG records are deleted. There must be no updates on any databases for the logs in question. Before IMS 13, the **DELETE.LOG** command always used the log Start Time. In IMS 13, DBRC checks to see whether the Stop Time is beyond the LOGRET time to determine whether the PRILOG or SECLOG should be deleted.

The benefit of this change is that PRILOG records that are closed but still within the retention period are retained until the retention period is passed.

5

IMS catalog

The IMS catalog is an optional system database (available since IMS 12) that stores metadata about your databases and applications. Its comprehensive view of IMS database metadata, fully managed by IMS, with which IMS can participate in solutions that require the exchange of metadata. A type of solution that requires such an exchange is business impact analysis. IMS 13 database versioning, Native SQL, and IMS Data Provider for .NET access to IMS Database all require the catalog.

The IMS Universal drivers are enhanced to use the IMS catalog.

This chapter describes the IBM IMS catalog and the way it can be used to expand and consolidate the information about IMS databases and their metadata.

This chapter also describes the IMS catalog, a trusted online source for IBM IMS database and application metadata information, and its usage by IMS Open Database and IMS Explorer. This chapter targets application developers who can benefit from these simplification and integration functions when they are facing any large-scale deployment of the IMS Open Database solution.

This chapter includes the following topics:

- Overview and objectives of the catalog
- Physical structure of the catalog database
- IMS catalog database installation and management
- Application usage of the catalog
- Enhancements to the IMS Universal drivers

5.1 Overview and objectives of the catalog

Before the introduction of the IMS catalog, information about the structure of the Data Language/I (DL/I) databases was spread across the following different data structures:

- The database description

  The *database description* (DBD) defines the characteristics of a database. For example, the DBD defines characteristics, such as its organization and access method, the segments and fields in a database record, and the relationship between the types of segments. Usually, information for the segments is limited to the few fields that are required to identify and search for segments across the hierarchical structures, that is, limited to the fields that are used as search arguments in the segment search argument (SSA), and the fields that are required to define logical relationships and secondary indexes.

- The COBOL copybooks and PL/I or C include members

  The details of the fields in each segment of the database are often defined in a Common Business Oriented Language (COBOL) copybook and in PL/I or C include members. These members detail all of the fields in each database segment (not just the fields that are used in SSAs), and are included into the source program when the program is compiled.

- The program specification block

  A *program specification block* (PSB) is used to define two things: the view of the databases to be used by a program, and any logical message destinations. These views are called *program communication blocks* (PCBs) because they are the means of communicating between the application program and IMS. There can be many PCBs in a PSB (as shown in Figure 5-1), with which a program can communicate with (access) multiple IMS databases. For each database that the program can access, a PCB is needed that describes the segments and the hierarchical structure that the program can access. The description can also indicate the sensitivity that a program has to the information; that is, whether the program can see only a subset of the segment types in the database, and a subset of the fields in these segments.

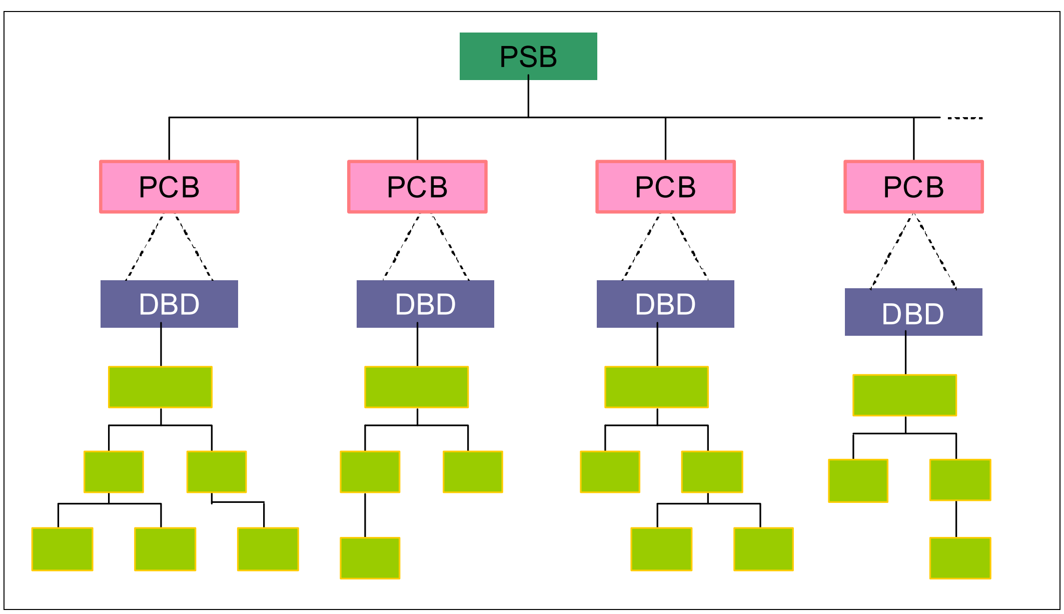


Figure 5-1 A PSB with multiple PCBs

A PCB can also allow a program to use different access paths through a database. The program can access a database through a secondary index or a logical relationship. The program view of the hierarchical structure of the database can be different from the hierarchical structure that is defined in the DBD, as shown in Figure 5-1 on page 136.

For online IMS systems, *application control block* (ACB) libraries contain the metadata that is used by IMS, which is created by performing an ACB generation from the DBD and PSB libraries. The system that uses DBD, PSB, and ACB libraries is proprietary. Typically, a DBD defines only a subset of the fields in a database record, such as the sequence (key) fields and any fields that are needed by a secondary index. Other fields in a record are typically defined only in a COBOL copybook or in a PL/I or C include file that is used by an application program.

The introduction of Java access to IMS data from JBP or JMP regions, or from remote systems or clients, requires easily accessible metadata. To provide this access, the contents of ACBLIB are offered as a Java class.

With the move to making IMS data more widely and easily accessible outside of the mainframe, application programmers must have easy and consistent access to the metadata. Since IMS V8, this data is provided by the DLIMODEL utility. This utility generates Java class files, which contain a static definition of the database design at the point when the DLIMODEL utility is run.

The drawback of the use of the DLIMODEL utility is that it can be a challenge to manage. Also, change control is needed when the underlying database definition changes. This method potentially allows the creation of multiple copies of the database metadata, each one of which must be updated with any database structure change.

There is a requirement for a source of metadata that is easier to manage and can be trusted to reflect the possible changes of the design of IMS databases. The metadata can also be defined in an open format. IMS Explorer for Development is designated as the strategic tool to create the metadata.

IMS 12 introduced the IMS catalog. The IMS catalog contains the metadata for databases and PSBs. It is accessed by using the Java Database Connectivity (JDBC) drivers and is also available to any tool or application. IMS metadata is available in an Extensible Markup Language (XML) format, and to standard DL/I applications in a traditional IMS segment format.

When the metadata is updated, the catalog is also updated to reflect the change; therefore, the data is always current. The data is current because the metadata is accessed dynamically from the active IMS catalog *high availability large database* (HALDB), rather than from static class files.

The catalog in IMS also includes a versioning system. This system allows the current version of the metadata and a user-specified number of previous versions to be kept and available for the applications.

The IMS catalog is a *partitioned hierarchical indexed direct access method* (PHIDAM) database that contains trusted metadata for IMS databases and applications. The *IMS Catalog Populate* utility (DFS3PU00) can optionally be used to initially populate the catalog by reading an existing ACBLIB and loading the available metadata into the catalog. All the information that is contained in the ACBLIB run time (which was derived from the DBDs and PSBs) is available to the users in the IMS catalog. Enhancements to the existing DBDGEN, PSBGEN, and ACBGEN processes allow database and application metadata to be defined to IMS.

The new *ACB Generation and Catalog Populate* utility (DFS3UACB) automatically updates the IMS catalog when the ACB members are generated. This update keeps the catalog always in a trusted state after initialization. The IMS catalog is the single, authoritative source of database and application metadata for all client applications. The catalog can also contain application metadata, such as decimal data specifications (scale and precision), data structure definitions, and mapping information.

The IMS catalog uses time stamps to identify the version of the metadata for each database, and can store multiple versions of metadata for each database. IMS provides the facility to keep a number of generations of this metadata and remove old metadata, according to a maximum number of generations or longevity that you specify. By default, the IMS catalog contains information for each DBD and PSB in the active IMS system ACBLIB. The IMS Catalog Record Purge utility (DFS3PU10) must be run to remove the extra definitions in the catalog.

5.2 Physical structure of the catalog database

The IMS catalog is a HALDB database that is a partitioned, IMS full-function database type. Before loading records into an IMS catalog, you must define the partitions to IMS. The IMS catalog is composed of the following objects:

- Primary database DFSCD000

  The access method is *overflow sequential access method* (OSAM), which is composed of the following database data sets:

  - Primary index data set
  - Indirect list data set (ILDS)
  - Four data set groups for the segments of the IMS catalog records

- Secondary index DFSCX000

  This index provides a cross-reference between the PSB segment and the DBD it is used to access. The data set for the secondary index database is a partitioned secondary index (PSINDEX).

The data that is stored in the IMS catalog includes all the metadata that was traditionally held in the DBD and PSB libraries. The IMS catalog also includes extra metadata information that is not available in the earlier releases of the DBD and PSB libraries. The IMS Explorer for Development is used to extend the metadata in the DBD, which is stored in the catalog. For more information about t about how the drivers use the metadata, see 5.5, “Enhancements to the IMS Universal drivers” on page 164.

5.2.1 Segments of the catalog database

The following segments are part of the catalog:

- Resource header (item name and type) for each DBD and PSB
- DBD resources:
  - Database structure definitions (ACCESS, RMNAME, and so on)
  - Data capture parameters
  - Physical database data set (DATASET) or area (AREA) definitions
  - Segment definitions (SEGM)
  - Field definitions (FIELD)
  - Marshaller definitions (DFSMARSH)
  - Logical children (LCHILD)

– Indexed field (XDFLD)
– Map definitions (DFSMAP)
– Case definitions (DFSCASE)
– Case field definition
– Case marshaller definition

- PSB resources:
  – Program Communication Block (PCB)
  – Sensitive segments (SENSEG)
  – Sensitive fields (SENFLD)
  – DBD cross-reference

There are several new macros in a DBD that with which IMS can better map the application data structures. They are all an extension of the field macro.

For each macro that used in a DBD or PSB, there is an equivalent segment in the IMS catalog database. The definition of the IMS catalog database can be found on the IMS.SDFSSRC library in members DFSCD000 (the database) and DFSCX000 (the secondary index).

Figure 5-2 shows the database structure for the IMS catalog database. To see the complete picture of the catalog database, import the catalog database DBD (DFSCD000) into IMS Explorer. The source of this DBD is available in the SDFSSRC data set, which was delivered with IMS since Version 12. For more information about instructions for importing DBDs into IMS Explorer, see 11.3, “Using IMS Explorer to capture IMS metadata” on page 328.

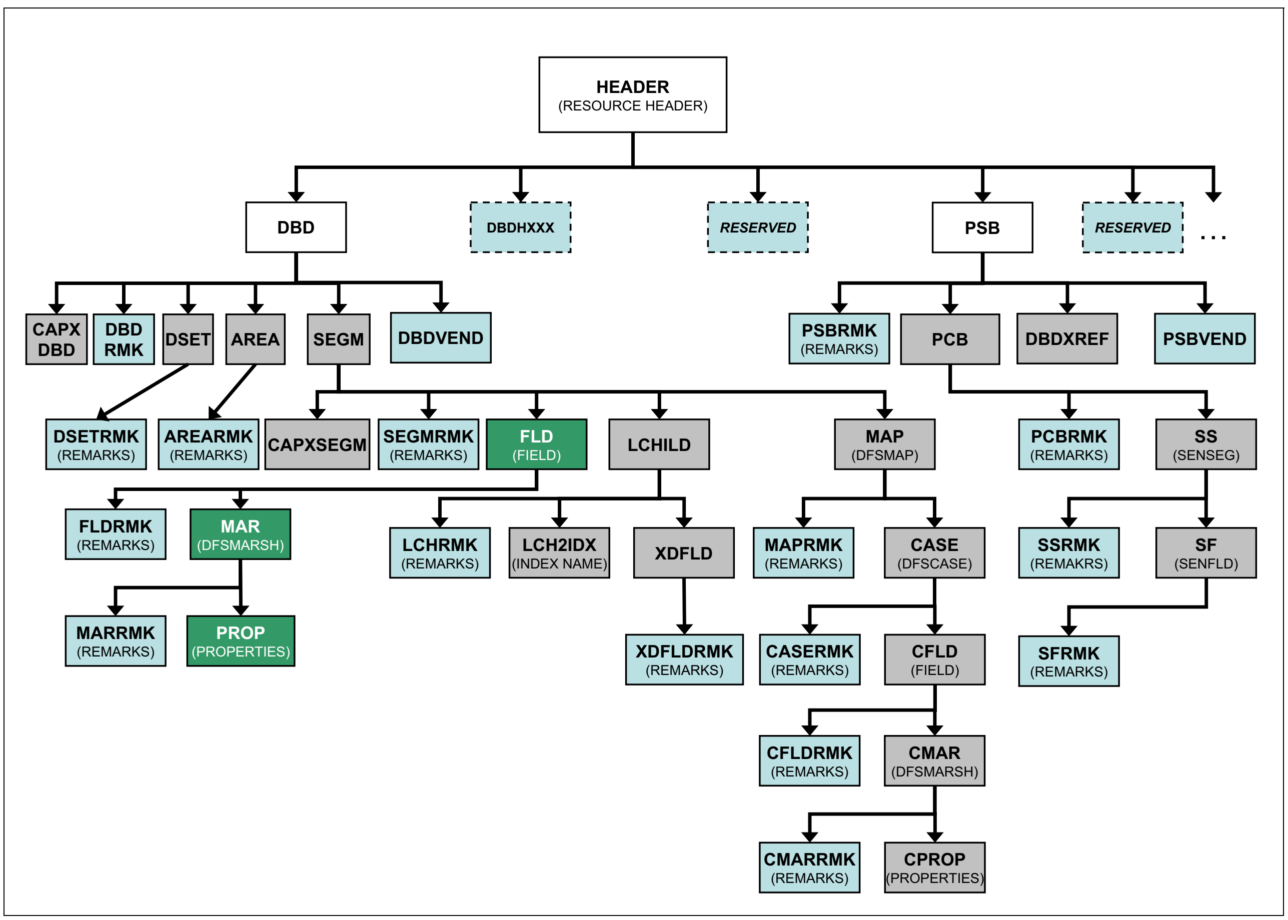


Figure 5-2 Database structure for the catalog database DFSCD000

The following SBs are supplied for accessing and updating the catalog database:

- DFSCPL00

  It is used for an initial load of the database (PROCOPT=L).

- DFSCP000

  It has read-only access (PROCOPT=G) for all segments. This PSB can be used for Assembler and COBOL programs for direct read-only access to the catalog database.

- DFSCP001

  It has update access (PROCOPT=A) for all segments. This PSB can be used for Assembler and COBOL programs for direct update access to the catalog database.

- DFSCP002

  It has read-only access (PROCOPT=G) for all segments. This PSB is used for PL/I programs for direct read-only access to the catalog database.

- DFSCP003

  It has read-only access (PROCOPT=G) for all segments. This PSB is used for PASCAL programs for direct read-only access to the catalog database.

There are several segments in the catalog database, including DBDVEND and PSBVEND, in the database definition for IBM tools and vendor products to store more information. There are also several reserved segments in the database to allow for future expansion.

Most of the segment types (including DBD, DSET, AREA, SEGM, FLD, LCHILD, and XFLD) contain metadata about the specific element that they represent. The data also includes the name of the segment that corresponds to the metadata that is stored.

Other segments (DBDRMK, DSETRMK, AREARMK, SEGMRMK, MAPRMK, and CASERMK) contain remarks about the element definition.

The CAPXDBD and CAPXSEGM segments contain information about Data Capture exit routines that are used.

5.3 IMS catalog database installation and management

The IMS catalog is implemented as a HALDB database. IMS normally requires each HALDB database to be registered in the Database Recovery Control (DBRC). There are reasons for some installations to choose not to use DBRC in some of their environments, especially when there are numerous test environments. Therefore, a method is provided to define the IMS catalog without the need to register the database in the RECON.

This section describes the following tasks for installation and management of the IMS catalog:

- Installation
- IMS catalog initial data population
- ACB generation and changes
- IMS Catalog Copy utility
- Keeping multiple instances of metadata in the catalog
- IMS Catalog Record Purge utility
- Automatically creating the IMS catalog database data sets
- Using the IMS catalog without DBRC
- Aliases and sharing
- Needed definitions for the IMS catalog

IMS provides several new utilities to maintain the catalog database when databases or PSB definitions are changed as part of your normal application development lifecycle.

5.3.1 Installation

The first step of the installation of the IMS catalog is to copy the supplied catalog DBD and PSB members from the SDFSRESL data set to your own DBD and PSB libraries. Example 5-1 shows the JCL to perform this copy.

Example 5-1 Copy the supplied DBD and PSB members to your own libraries

```
//CPYCMEM EXEC PGM=IEBCOPY
//SDFSRESL DD  DSN=IMS.SDFSRESL,DISP=SHR
//DBDLIB   DD  DSN=IMS.DBDLIB,DISP=SHR
//PSBLIB   DD  DSN=IMS.PSBLIB,DISP=SHR
//SYSIN    DD  *
          COPY   OUTDD=DBDLIB,INDD=((SDFSRESL,R))
          SELECT MEMBER=(DFSCD000,DFSCX000)
          COPY   OUTDD=PSBLIB,INDD=((SDFSRESL,R))
          SELECT MEMBER=(DFSCPL00,DFSCP000,DFSCP001,DFSCP002,DFSCP003)
```

The source for the catalog DBD and PSBs is shipped for reference in the IMS source library SDFSSRC. However, the object code in SDFSRESL must be used for execution.

After the DBDs and PSBs are copied to your libraries, you must build them as ACBs by using the traditional ACB process, as shown in Example 5-2.

Example 5-2 ACBGEN for IMS catalog

```
// JCLLIB ORDER=IMS.PROCLIB
// EXEC ACBGEN
//SYSIN    DD  *
    BUILD PSB=(DFSCPL00)
    BUILD PSB=(DFSCP001)
    BUILD PSB=(DFSCP000)
    BUILD PSB=(DFSCP002)
    BUILD PSB=(DFSCP003)
```

Next, create the catalog database data sets. The space that is allocated for the catalog database must be sufficient to store the number of DBD and PSB definitions for your environment. This creation can be done in the following ways:

- The database data sets can be allocated with the *job control language* (JCL) similar to what is shown in Example 5-3 on page 142.
- The IMS Catalog Partition Definition Data Set utility (DFS3UCD0) can be used to create the catalog partition definition database data set if DBRC is not in use. It also creates the database data sets for the catalog database.
- The IMS Catalog Populate utility (DFS3PU00) is used to load or insert records into the catalog database, whether DBRC is used for the catalog database. If any of the database data sets do not exist, the utility creates them. The size parameters that are used for automatic creation are specified in the catalog section of the DFSDFxxx procedure library (PROCLIB) member.

Example 5-3 on page 142 shows the process of defining the database data sets and indexes.

Example 5-3 Define the HALDB data sets, the ILDS, and the primary and secondary indexes

```
//PHIDAM EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
   ALLOCATE DSNAME('IMS.DFSCDI2D.A00001') FILE(A00001) -
       RECFM(F,B,S) DSORG(PS) NEW CATALOG -
       SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

   ALLOCATE DSNAME('IMS.DFSCDI2D.B00001') FILE(B00001) -
       RECFM(F,B,S) DSORG(PS) NEW CATALOG -
       SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

   ALLOCATE DSNAME('IMS.DFSCDI2D.C00001') FILE(C00001) -
       RECFM(F,B,S) DSORG(PS) NEW CATALOG -
       SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

   ALLOCATE DSNAME('IMS.DFSCDI2D.D00001') FILE(D00001) -
       RECFM(F,B,S) DSORG(PS) NEW CATALOG -
       SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

   DEFINE CLUSTER(NAME('IMS.DFSCDI2D.L00001') -
      FREESPACE(80 10) SHAREOPTIONS(3 3) KEYS(9 0) -
      RECORDSIZE(50 50) SPEED CYLINDERS(1,1) VOLUMES(SBOXI5)) -
      DATA(CONTROLINTERVALSIZE(4096)) -
      INDEX(CONTROLINTERVALSIZE(2048))

   DEFINE CLUSTER(NAME('IMS.DFSCDI2D.X00001') INDEXED -
      KEYS(16,5) VOL(SBOXI5) REUSE RECORDSIZE (22,22)) -
      DATA(CONTROLINTERVALSIZE(4096))

//PSINDEX EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
  DEFINE CLUSTER (NAME('IMS.DFSCXI2D.A00001') INDEXED -
     SHAREOPTIONS(3 3) KEYS(37,45) REUSE RECORDS(5,5) VOL(SBOXI5) -
     RECORDSIZE(82,82)) DATA(CONTROLINTERVALSIZE(4096))
```

After the database data sets are created, you must update the IMS.PROCLIB(DFSDFxxx) member to define the catalog parameters for your IMS system. There are two sections to be added to the member, depending on which of the following configurations you choose to use:

- A unique IMS catalog for each system
- A unique DFSDFxxx member for each system
- A shared IMS catalog
- A shared DFSDFxxx member

Example 5-4 shows the simplest environment, which is a single IMS with a single catalog prefixed with the default name DFSC.

Example 5-4 IMS.PROCLIB(DFSDFxxx) for a single IMS system

```
*--------------------------------------------------------------------*
* IMS CATALOG SECTION                                                *
*--------------------------------------------------------------------*
<SECTION=CATALOG>
  CATALOG=Y
  ALIAS=DFSC
```

Example 5-5 shows a shared DFSDFxxx member with a separate IMS catalog for each IMS system (defined with an ALIAS that matches the IMSID). For more information about the use of aliases for the IMS catalog, see 5.3.9, “Aliases and sharing” on page 149.

Example 5-5 IMS.PROCLIB(DFSDFxxx) for multiple IMS systems

```
*--------------------------------------------------------------------*
* IMS CATALOG SECTION for IMS I12B *
*--------------------------------------------------------------------*
<SECTION=CATALOGI12B>
  CATALOG=Y
  ALIAS=I12B
*--------------------------------------------------------------------*
* IMS CATALOG SECTION for IMS I12D *
*--------------------------------------------------------------------*
<SECTION=CATALOGI12D>
  CATALOG=Y
  ALIAS=I12D
```

The catalog can be activated by using the new DFSDFxxx member with a cold start, warm start, or even an emergency restart of the IMS system. The restart is required because IMS reads only the DFSDF member during initialization.

5.3.2 IMS catalog initial data population

After you define the IMS catalog database data sets, you must run a new utility to read the IMS ACBLIB to initially load (populate) the IMS catalog database. As is usual for all new releases of IMS, the ACBLIB must be rebuilt from the DBD and PSB libraries. This rebuild is done by using the ACBGEN utility for the new release before the ACBLIB is used, including populating the catalog. The catalog populate process must be done before an IMS system can start using the catalog. An initial load of the IMS Catalog removes any existing data from the catalog before the new definitions are loaded from the ACB library.

You start by defining the IMS catalog database to DBRC or with the non-DBRC utility. When DBRC is used, this process can be done with the batch DBRC utility (DSPURX00) by using the **INIT.DB** and **INIT.PART** commands for the catalog database. The process to populate the catalog also can be done by using the IMS Partition Definition Utility (PDU) application in the Time Sharing Option (TSO).

DBRC registration is optional. For more information about when DBRC is not used, see 5.3.8, “Using the IMS catalog without DBRC” on page 148.

The initial data population of the IMS catalog is done by using the *IMS Catalog Populate* utility (DFS3PU00), as shown in Figure 5-3.

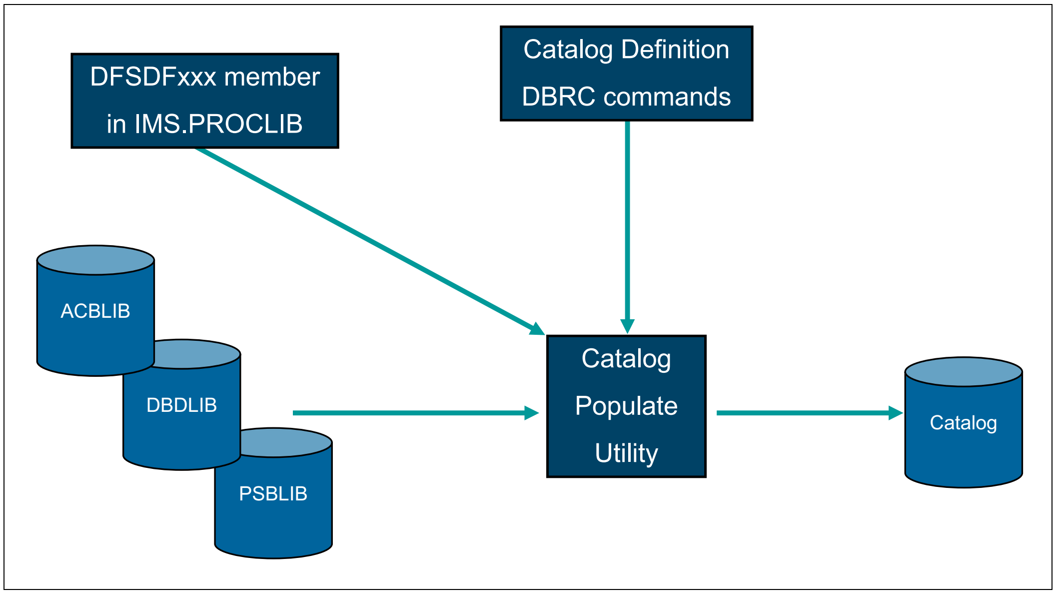


Figure 5-3 IMS Catalog Populate utility (DFS3PU00)

The utility reads the IMS.PROCLIB(DFSDFxxx) member, then builds the IMS catalog from the ACB libraries that are referenced in the JCL. The utility also reads the DBD and PSB libraries to capture the information about any generalized sequential access method (GSAM) DBDs. This method is used because they are not built as ACB library members.

After execution, a HALDB REORG record is recorded in the RECON.

By using the IMS Explorer, we can add application metadata to these database definitions. This process is needed because at this stage the IMS catalog holds only the rudimentary information that is available from the DBD and PSB definitions. For more information about this process, see 5.5, “Enhancements to the IMS Universal drivers” on page 164.

Example 5-6 shows the JCL that is used to populate an IMS catalog from the current ACB libraries.

Example 5-6 Running the IMS Catalog Populate utility (DFS3PU00)

```
//LOADCAT EXEC PGM=DFS3PU00,
// PARM=(DLI,DFS3PU00,DFSCPL00,,,,,,,,,,,Y,N,,,,,,,,,,,,'DFSDF=12D')
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//IEFRDER  DD  DISP=(,CATLG),DSN=IMS.PU000.LOG(+1),
// SPACE=(TRK,(5,5),RLSE),UNIT=SYSALLDA
//DFSVSAMP DD  DSN=IMS.PROCLIB(DFSVSMDB),DISP=SHR
//IMSACB01 DD  DSN=IMS.ACBLIBA,DISP=SHR
//IMSACB02 DD  DSN=IMS.ACBLIBB.DOPT,DISP=SHR
```

5.3.3 ACB generation and changes

After you migrate to an IMS system that is using an IMS catalog, you must keep the ACB libraries and the catalog synchronized. IMS uses time stamps to ensure consistency. With time stamps, you can ensure that the catalog data can always be trusted as an accurate equivalent of the metadata that is held in the DBD, PSB, and ACB libraries.

There is a new *ACB Generation and IMS Catalog Populate* utility (DFS3UACB). This utility is used to generate ACB members in an IMS.ACBLIB data set and create the corresponding metadata records in the IMS catalog in a single job step, as shown in Figure 5-4. The DFS3UACB utility can be used in load mode to initially populate the catalog, or in update mode to add a version of the new or changed definitions. In update mode, a new version of any changed definitions is created in the catalog, rather than altering the existing definitions in the catalog.

When the utility is run in load mode, all existing records in the IMS catalog are discarded.

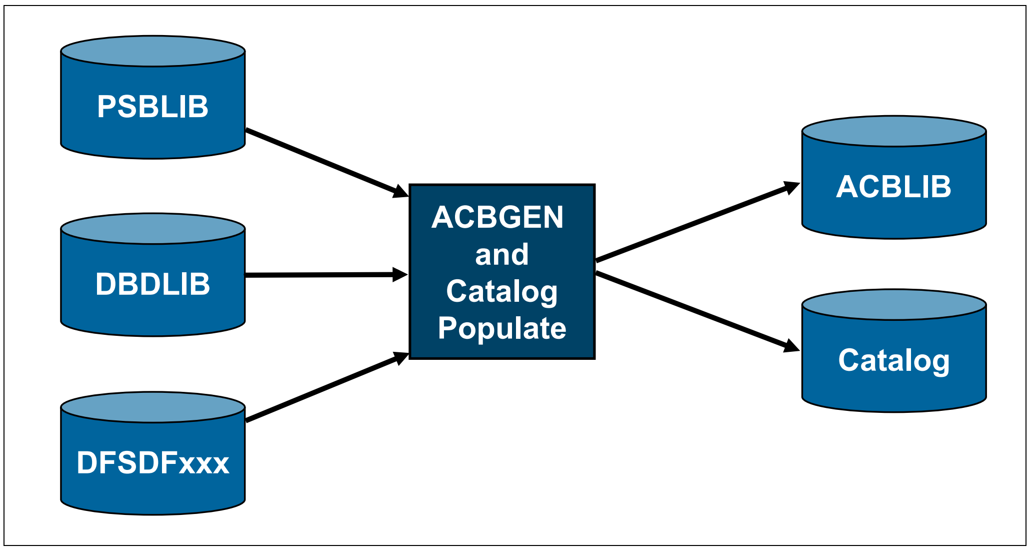


Figure 5-4 ACB Generation and IMS Catalog Populate utility

The new ACB Generation utility writes logs. If the IMS catalog is defined in DBRC, it updates the RECON. It can run as a batch message program (BMP) or DLIBATCH mode. If it is run as a BMP, ensure that the catalog database is opened by IMS for update access. Usually, IMS opens the catalog database in read-only mode.

5.3.4 IMS Catalog Copy utility

After you migrate to an IMS catalog, you must ensure that the IMS catalog is updated whenever you update the IMS ACB library. When you migrate applications from one environment to another (for example, from the test through to production environments), the new *IMS Catalog Copy* utility can be used to keep the metadata synchronized.

This utility allows the export and import of metadata information between catalogs.

The utility DFS3CCE0 exports from a catalog and the ACBLIB, DBDLIB, and PSBLIB libraries. This utility copies an IMS catalog and any included ACB, DBD, and PSB libraries to export data sets in the same job step.

The utility DFS3CCI0 imports from the export data set into another catalog, ACBLIB, DBDLIB, and PSBLIB libraries.

At the destination environment, the import module DFS3CCI0 loads or updates an IMS catalog. The module also copies any included ACB, DBD, and PSB libraries from the export data sets into their destination data sets.

For the import function of the IMS Catalog Copy utility, the primary input to the utility is a data set that contains the new copy of the IMS catalog. A CCUCATIM data definition (DD) statement is required to identify this data set.

If the ACB libraries, DBD libraries, and PSB libraries were copied during the export function, they are identified in the import JCL by the following DD statements:

- **`CCUACBIM DD`** statement for the ACB library export data set
- **`CCUDBDIM DD`** statement for the DBD library export data set
- **`CCUPSBIM DD`** statement for the PSB library export data set

The new utilities ensure that the updates to the ACB libraries result in parallel updates to the IMS catalog.

Optionally, the IMS Catalog Copy utility can also create copies of the ACB, DBD, and PSB library members that correspond to the IMS catalog records that are copied.

The IMS Catalog Copy utility creates an import statistics report for the record segments to be loaded or updated in the IMS catalog. When the Catalog Copy utility runs in analysis-only mode, the report reflects only certain statistics. The report reflects only the potential statistics if the IMS catalog were loaded or updated from the ACB libraries that are currently used as input to the utility.

The IMS Catalog Copy utility can be run as a batch job or as an online BMP.

5.3.5 Keeping multiple instances of metadata in the catalog

The catalog can hold multiple instances of the DBD and PSB metadata. The information in the IMS catalog is time-stamped. IMS uses these time stamps to keep multiple instances of the metadata.

The catalog section of the DFSDFxxx PROCLIB member specifies the IMS-wide default values for the retention of metadata in the catalog. The parameters define the maximum number of instances and the minimum retention period for which information is to be kept in the catalog. This specification is similar to the way that information is stored by DBRC in the RECON to keep records of a number of Image Copy sets. However, the information is not automatically purged, as it is with DBRC. The DBDs and PSBs are deleted by the *IMS Catalog Record Purge* utility (DFS3PU10) only. The retention parameters for specific databases can be specified with DFS3PU10.

The following two parameters are used to control the optional **RETENTION** statement in the DFSDFxxx catalog section: INSTANCES=nnn and DAYS=ddd:

- INSTANCES defines the maximum number of instances of a DBD or PSB to be kept in the IMS catalog database. The value can be 1 - 65535; the default value is 2. When the maximum value is reached, the oldest instance (based on the ACBGEN time stamp) is replaced by the newest instance.
- DAYS defines the minimum number of days an instance remains in the catalog. The value can be 0 - 65535; the default value is 0 (function disabled). When an instance of the catalog metadata is older that the specified instance, it becomes a candidate for removal. It might be removed when new instances of the same DBD or PSB are added to the IMS catalog.

These parameters work in the same way that GENMAX and RECOVPD work with DBRC. If the maximum number of versions is reached but the retention period did not expire, a new catalog record is kept. The oldest record is not removed.

Example 5-7 shows the specification in the IMS.PROCLIB(DFSDFxxx) member (where at least five generations of metadata are kept) for a minimum of 30 days.

Example 5-7 Setting default maximum generations and retention periods for catalog metadata

```
<SECTION=CATALOG>
CATALOG=Y
ALIAS=DFSC
RETENTION=(INSTANCES=5,DAYS=30)
```

When multiple instances are stored in the catalog, the catalog might still have the previous instance available if an online change is reversed.

For logically related databases, the Catalog Populate utility sets the time stamp to zero (the DBD information was kept only in the DBDLIB and not the ACBLIB in the past). The time stamp is set to zero until the combined ACBGEN/Populate utility runs for those databases.

Metadata information in the catalog can be removed by the IMS Catalog Record Purge utility (DFS3PU10) if it is outdated or if the number of instances exceeds the specified retention value. When DFS3PU10 is used, the default of two instances applies. If no retention value is specified, each execution of ACBGEN or Populate Utility adds an instance.

5.3.6 IMS Catalog Record Purge utility

The IMS Catalog Record Purge utility (DFS3PU10) provides the following functions:

- Sets the record retention criteria for specific DBD and PSB records in the catalog database.
- Produces a list of DBDs and PSBs with versions that are no longer needed according to the current retention criteria for each record.
- Purges specific instances (based on the ACB time stamp) of a record for a DBD or PSB resource from the IMS catalog database.
- Purges unnecessary instances of IMS catalog records from the catalog database that is based on criteria that you specify.

5.3.7 Automatically creating the IMS catalog database data sets

Use the DFSDFxxx member of the IMS PROCLIB data set to specify processing options for the IMS catalog. The DFSDFxxx member contains parameters that are organized into sections. Example 5-8 shows the specify options for the IMS catalog. In a data sharing environment, this section defines the IMS catalog for all IMS systems that are not individually configured with a CATALOGxxxx section.

Example 5-8 IMS catalog parameters in DFSDFxxx

```
<SECTION=CATALOG>
CATALOG=Y
ALIAS=DFSC
/* IXVOLSER=vvvvvv if DFSMS is not used */
DATACLAS=IMSDATA
MGMTCLAS=EXTRABAK
STORCLAS=IMS
SPACEALLOC=500
```

By using the other parameters, you can define the DFSMS DATALCLAS, MGMTCLAS, STORCLAS, or volume serial number if DFSMS is not used.

SPACEALLOC=nnnn (0 - 9999; the default is 500) is used to define the space as a percentage of the predefined IMS value.

5.3.8 Using the IMS catalog without DBRC

The IMS catalog database is a standard PHIDAM database. That type of database often requires registration to DBRC. Among other functions, DBRC tracks the logs that are used when the database is updated, and can be used to generate jobs to perform image copies and accumulate changes with the IMS Change Accumulation utility.

You might be running your IMS system with non-HALDB databases that are not registered to DBRC (which is a common configuration for test systems). If the databases are not registered, IMS does not insist that the catalog HALDB database (and index) is registered in RECON. If the HALDB is not registered, IMS does not track logs, change accumulations, or image copies.

The backup, logging, and recovery of the IMS catalog is the responsibility of the user. This process can be done by running stand-alone IBM MVS utilities or by DFSMShsm backup and recovery. In some cases, the usage of the IMS Catalog Populate utility to rebuild the database might be the best choice.

To overcome the need for IMS to have the IMS catalog PHIDAM database metadata (normally stored in the DBRC RECON), IMS has special handling for the IMS catalog. Instead of having database, partition, and database data set records that are defined in the RECON, IMS uses dynamic allocation (the DFSMDA macro) with the **TYPE=CATDBDEF** statement. This statement defines the dynamic allocation parameter list for the IMS catalog partition definition data set. This data set contains the definitions for the catalog HALDBs that are not defined in the DBRC RECON data set. The DD name of the catalog partition definition data set is DFSHDBSC.

A new utility, DFS3UCD0, is used to create the HALDB partition definition data set. You must supply a system input stream (SYSIN) data set to define the data that normally is used on the **INIT.DB** and **INIT.PART** statements when a database is registered in DBRC, as shown in Example 5-9. The structure information is stored in the definition data set (DFSHDBSC).

Example 5-9 Example SYSIN for DFS3UCD0

```
HALDB=(NAME=DFSCD000,HIKEY=YES)

PART=(NAME=DFSCD000,
    PART=DFSCD01,
    DSNPREFX=IMSTESTS.DFSCD000,
    KEYSTHEX=FFFFFFFFFFFFFFFFFFFFFFFFFFFFF)

HALDB=(NAME=DFSCX000,HIKEY=YES)

PART=(NAME=DFSCX000,
    PART=DFSCX01,
    DSNPREFX(IMSTESTS.DFSCX000)
    KEYSTHEX=FFFFFFFFFFFFFFFFFFFFFFFFFFFFF)
```

You then must add is a statement in the IMS.PROCLIB(DFSDFxxx) member to define the IMS catalog that is not registered in DBRC, as shown in Example 5-10.

Example 5-10 IMS.PROCLIB(DFSDFxxx) definition for a non-DBRC IMS catalog

```
<SECTION=DATABASE>
UNREGCATLG=DFSCD000
```

Optionally, to avoid making JCL changes, you can build a DFSMDA dynamic allocation macro to define the DFSHDBSC to IMS, as shown in Example 5-11. **TYPE=CATDBDEF** is a new specification for an unregistered IMS catalog database only.

Example 5-11 DFSMDA definition

```
        DFSMDA TYPE=INITIAL
         DFSMDA TYPE=CATDBDEF,DSNAME=IMS.IMS13D.CATALOG.DEFDS
         DFSMDA TYPE=FINAL
         END
```

5.3.9 Aliases and sharing

In an IMSplex, you have several options for how you set up the ACBLIB and the IMS catalog. You can share or clone the ACB libraries ACBLIBA or ACBLIBB. Similarly, the IMS catalog can be shared, or it can use a separate IMS catalog for each IMS system.

The way that ACB libraries in IMS V10 or IMS V11 are used depends on whether you are using local, global, and member online changes. IMS V12 and later supports these configuration options, and you can configure the IMS catalog to participate with your existing shared or discrete ACB libraries. The IMS catalog can be shared between multiple IMS systems.

> **Tip:** Convert your systems to use global online change, which dynamically allocates ACB libraries and the usage of member online changes.

Figure 5-5 shows the migration of a system from an IMSplex where each IMS has its own cloned ACB library pair. However, a single shared IMS catalog also is introduced that is populated from all the existing ACB libraries.

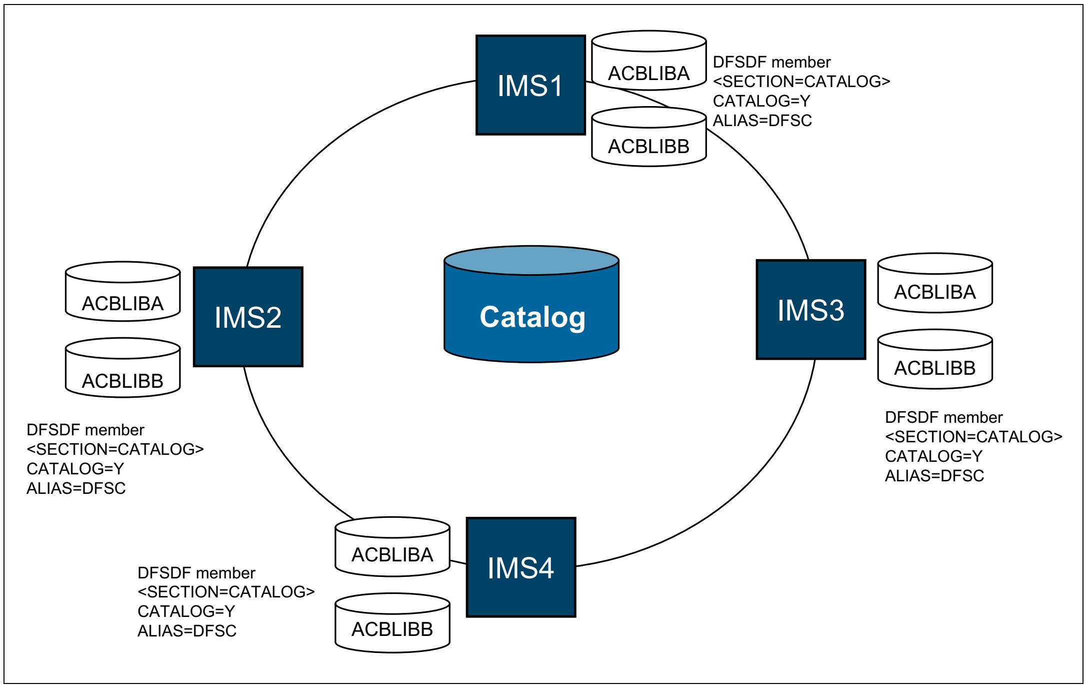


Figure 5-5 Multiple IMS, cloned ACBLIB, and shared IMS catalog

Figure 5-6 shows a system in which you start from separate (cloned) ACB libraries. In this example, a unique aliased IMS catalog was created for each IMS system.

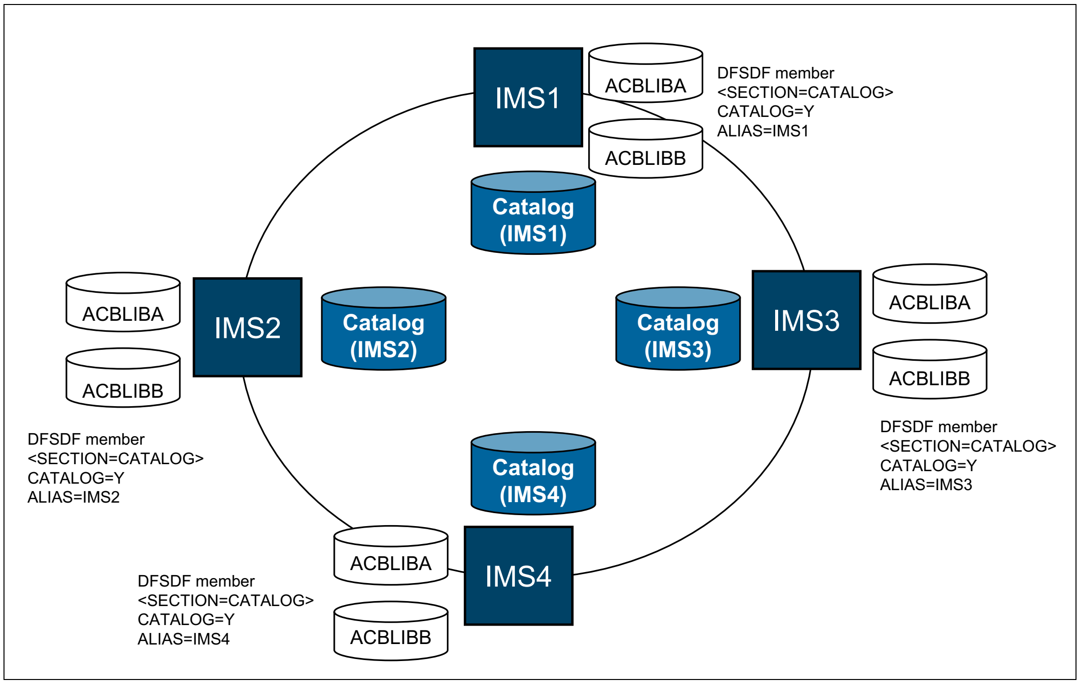


Figure 5-6 Multiple IMS, cloned ACBLIB, and one IMS catalog for each IMS

Figure 5-7 shows the integrated system in which there is one pair of ACB libraries for the IMSplex and a single shared IMS catalog (registered in DBRC with SHARELVL=3). In this system, global online change and member online change are used.

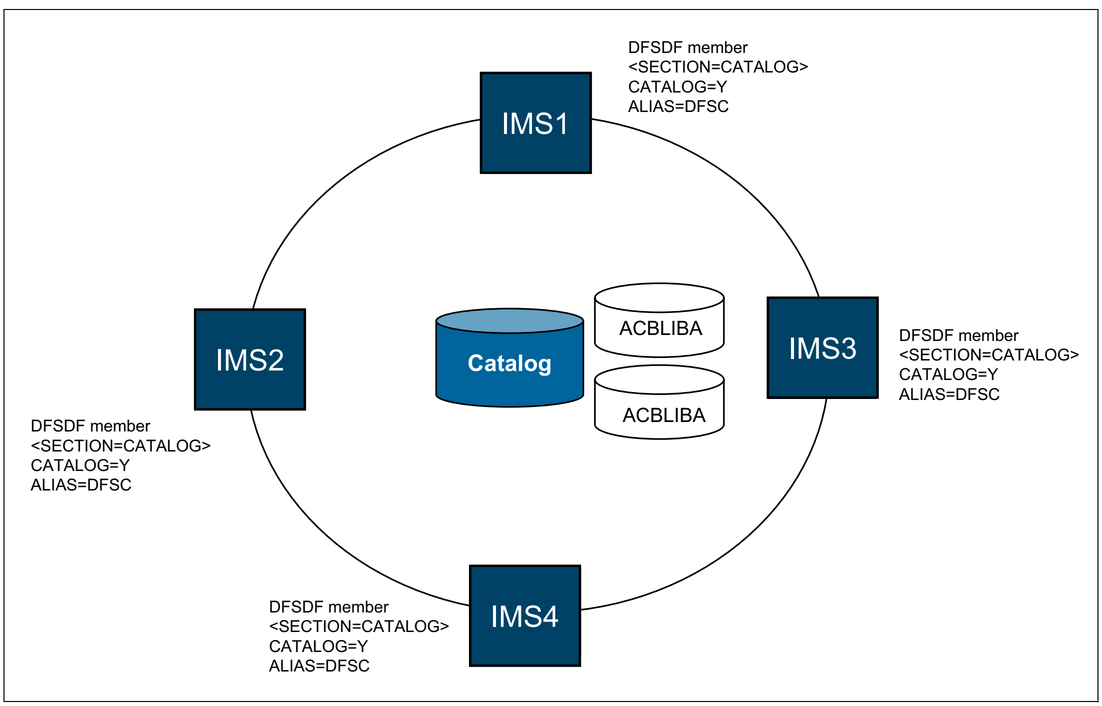


Figure 5-7 Multiple IMS, shared ACBLIBs, and shared IMS catalog

IMS handles the loading of the DBDs (DFSCD000 and DFSCX000) and the PSBs (DFSCP000, DFSCP001, DFSCP002, and DFSCP003). IMS also makes the internal changes that are needed when aliased names are used.

5.3.10 Needed definitions for the IMS catalog

Before you can create the IMS catalog and initially load it with the populate utility, you must update the IMS system to use it by adding statements to the IMS.PROCLIB(DFSDFxxx) member.

You have some flexibility in how to define the CATALOG section in the DFSDFxxx member. A simple single IMS is shown in Example 5-12 with its suggested ALIAS name.

Example 5-12 Single IMS system

```
<SECTION=CATALOG>
CATALOG=Y
ALIAS=DFSC
```

Alternatively, if you have two or more IMS systems that use shared queues, you might want them to share a single IMS catalog and the same DFSDFxxx proclib member. In this case, you can define different section suffixes in the **CATALOG** statement by using the IMS ID, and you can define the catalog database with a single shared ALIAS name, as shown in Example 5-13.

Example 5-13 Multiple IMS systems

```
<SECTION=CATALOGI12A>
CATALOG=Y
ALIAS=I12Q
<SECTION=CATALOGI12C>
CATALOG=Y
ALIAS=I12Q
```

5.4 Application usage of the catalog

IMS V12 and later can access the IMS catalog when IMS metadata is required. IMS also provides a user interface to the IMS catalog data and a new DL/I call to access the IMS catalog. One source of metadata for populating the catalog is the DBDs and PSBs. The extensions of the metadata information and the other data manipulation possibilities that are introduced require more information than was possible with the DBD and PSB sources before IMS V12. Therefore, several DBDGEN and PSBGEN macros were enhanced in IMS V12.

5.4.1 DBD and PSB source changes

There are new **DBDGEN** statements in IMS V12 and changes to existing statements. The new and changed statements extend the metadata information that is available to the catalog. Much of this metadata information is the basis for the extended Object usage by the Java language.

DFSMARSH statement in the DBD

The **DFSMARSH** statement in a DBD defines the marshalling attributes for field data. The **DFSMARSH** statement must immediately follow the **FIELD** statement to which it applies. You can use the **DFSMARSH** statement to specify the following data marshalling attributes for the data that is contained in a field:

- You can specify the code page or character encoding that defines the character data in a field.
- You can specify a data-conversion routine for IMS to use when field data is converted. For example, use it when you are converting field data from the data type that IMS uses to physically store data to a data type that is expected by an application program.
- You can specify whether a numeric data type is signed or not with the **ISSIGNED** parameter.
- The pattern to use for dates and times can be specified with the **PATTERN** parameter.
- You can specify the properties that are used with a user-provided data-conversion routine.

The MAR segment type in the catalog database contains information about a field marshaller definition in an IMS database. Each FLD segment can have a MAR child segment that contains data marshalling properties for that field. The following information in this segment type is generated from the input parameters of the **DFSMARSH** statement of the DBDGEN utility:

- **ENCODING**

  Specifies the default encoding of all character data in the database that is defined by this DBD.

  The default **ENCODING** is CP1047 (EBCDIC). This default value can be overridden for individual segments and fields.

- **USERTYPECONVERTER**

  Specifies the fully qualified Java class name of the user-provided Java class to be used for type conversion.

- **INTERNALTYPECONVERTER**

  Specifies the internal conversion routine that IMS uses to convert the IMS data into Java objects for Java application programs. IMS requires the specification of either **INTERNALTYPECONVERTER** or **USERTYPECONVERTER**, but not both. Valid values for the **INTERNALTYPECONVERTER** parameter include:

  `ARRAY`, `BINARY`, `BIT`, `BLOB`, `BYTE`, `CHAR`, `CLOB`, `DOUBLE`, `FLOAT`, `INT`, `LONG`, `PACKEDDECIMAL`, `SHORT`, `STRUCT`, `UBYTE`, `USHORT`, `UINT`, `ULONG`, `XML_CLOB`, and `ZONEDDECIMAL`.

- **ISSIGNED**

  Valid only for `DATATYPE=DECIMAL`. Valid values are `Y` (default) or `N`.

- **PROPERTIES**

  Specifies properties for user type converter names with the **USERTYPECONVERTER** parameter. These properties are passed to the user type converter.

- **PATTERN**

  An optional field that specifies the pattern to use for the date, time, and time stamp Java data types.

  The **PATTERN** parameter applies only when the `DATE`, `TIME`, or `TIMESTAMP` is specified on the **DATATYPE** keyword in the **FIELD** statement. Also, `CHAR` must be specified on the **INTERNALTYPECONVERTER** keyword in the **DFSMARSH** statement.

DFSMAP statement

The **DFSMAP** statement enables the alternative mapping of fields within a segment.

The **DFSMAP** statement defines a group of one or more map cases and relates the cases to a control field. The control field identifies which map case is used in a particular segment instance, as shown in Figure 5-8.

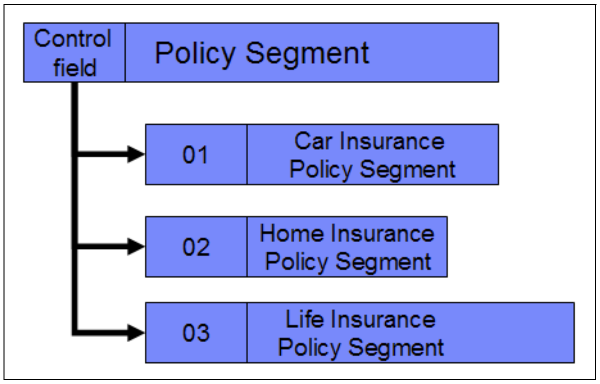


Figure 5-8 Several mappings for the same segment

The DFSMAP statement includes the following parameters:

- **NAME**

  Defines the name of this map.

- **DEPENDING ON**

  External name of the control field within this segment that contains the value that determines which map case is used for a particular segment instance. If the control field does not contain a value that corresponds to a **CASEID** in a **DFSCASE** statement for this map, the map is not used for this segment instance.

  If the **FIELD** statement that defines the control field does not explicitly code the **EXTERNALNAME** parameter, specify the value of the **NAME** parameter in the **DEPENDING ON** field.

- **REMARKS**

  Allow for a comment up to 256 characters long to be recorded in the DBD and the IMS catalog database.

DFSCASE statement

Many applications allow for several different layouts for the same segment type in an IMS database. In COBOL, this configuration is done with the **REDEFINES** statement and in PL/I with the **DEFINED** statement. The corresponding facility in IMS is the **DFSCASE** statement, which defines a map case; that is, a conditional mapping of a set of fields in a segment. Each map case has a name and an identifier (ID). A single segment can be defined with multiple map cases. The map cases within a segment are grouped by the **DFSMAP** statement, which also associates the map cases with their control field.

Each map case is defined with a case ID. The case ID is stored in the control field to identify which case map is used for a particular segment instance. Typically, the control field is defined at the beginning of the segment before the **DFSMAP** statement.

The fields that make up a map case identify the map case that they belong to by the name on the **CASENAME** parameter in the **FIELD** statement. **CASENAME** is valid and required only to associate a **FIELD** statement with the preceding **DFSCASE** statement that defines the map case to which this field belongs. The value of **CASENAME** must match the value that is specified on the **NAME** parameter of the **DFSCASE** statement.

The ID of a map case is specified on the **CASEID** parameter. In a segment instance, the following ID is inserted in a control field to indicate which map case is in use:

- **NAME**

  Defines the name of this case.

- **CASEIDTYPE**

  Defines the data type of the value that is specified in the **CASEID** parameter.

- **MAPNAME**

  The name of the map that this case belongs to, as specified on the **NAME** parameter in the **DFSMAP** statement.

- **CASEID**

  Defines a unique identifier for the case. A segment instance specifies the **CASEID** value in a user-defined control field when part or all of the field structure of the segment is mapped by this case. The **CASEID** must be unique within the map to which the case belongs.

- **REMARKS**

  Allows for a comment of up 256 characters to be recorded in the DBD and the IMS catalog database.

Enhancements to DBD definition statements

There are several enhancements to the statements that are used to describe a database definition with IMS V12.

Changes to the DBD statement

ENCODING specifies the default encoding of all character data in the database that is defined by this DBD; otherwise, the default **ENCODING** is CP1047 (EBCDIC). The value can be overridden in individual segments or fields.

Changes to the SEGM statement

The following changes were made to the SEGM statement:

- **ENCODING**

  Specifies the default encoding of all character data in the database that is defined by this DBD; otherwise, the default **ENCODING** is CP1047 (EBCDIC). The value can be overridden in individual fields.

- **EXTERNALNAME**

  This parameter is used to give a segment an extended (1 - 128 byte) name.

Changes to the FIELD statement

The following changes were made to the FIELD statement:

- **NAME**

  Specifies the name of this field within a segment type. The name must be provided to search on the field.

For key-sequenced field types and field types that are referenced by an **XDFLD** statement, the **NAME** parameter is required.

For other field types, you can optionally omit the **NAME** parameter when the **EXTERNALNAME** parameter is specified.

- **PARENT**

  Specifies the name of a field that is defined as a structure or array in which this field is contained. Must be the value of the **EXTERNALNAME** parameter in the definition of the referenced field.

  The referenced field must be defined with either `DATATYPE=ARRAY` or `DATATYPE=STRUCT`.

- **CASENAME**

  The name of the map case to which this field belongs.

  This parameter is required only to associate a field statement with the preceding **DFSCASE** statement that defines the map case.

  This parameter must match the value that is specified on the **NAME** parameter of the **DFSCASE** statement.

- **DATATYPE**

  This parameter is an optional value that specifies one of the following external data types for the field:

  - `ARRAY`
  - `BINARY`
  - `BIT`
  - `BYTE`
  - `CHAR`
  - `DATE`
  - `DECIMAL` (with precision and scale)
  - `DOUBLE`
  - `FLOAT`
  - `INT`
  - `LONG`
  - `OTHER`
  - `SHORT`
  - `STRUCT`
  - `TIME`
  - `TIMESTAMP`
  - `XML`

- **REDEFINES**

  This parameter that specifies the name of a field in the segment. The field must be the same length as the field that is redefined.

  The field cannot be an ARRAY or contain an ARRAY.

- **EXTERNALNAME**

  This parameter is an optional alias for the **NAME=** parameter. Java application programs use the external name to refer to the field.

  The **EXTERNALNAME** parameter is required only when the **NAME** parameter is not specified. If the **NAME** parameter is not specified, you cannot search on this field.

- **`DEPENDSON`**

  This parameter specifies the name of a field that defines the number of elements in a dynamic array. Referenced fields must precede the field statement that specifies this parameter.

  The name that is specified must be the value, whether explicitly defined or accepted by default, of the **`EXTERNALNAME`** parameter in the definition of the referenced field.

  The **`DEPENDSON`** parameter is valid only when `DATATYPE=ARRAY` is also specified.

  The field that is referenced by the **`DEPENDSON`** parameter must be defined with one of the following `DATATYPE` values:

  – `INT`
  – `SHORT`
  – `LONG`
  – `DECIMAL`

- **`MINOCCURS`**

  Valid for `DATATYPE=ARRAY` only, **`MINOCCURS`** is a required numeric value that specifies the minimum number of elements in an ARRAY. **`MINOCCURS`** is invalid for all other data types.

- **`MAXOCCURS`**

  Valid for `DATATYPE=ARRAY` only, **`MAXOCCURS`** is a required numeric value that specifies the maximum number of elements in an ARRAY.

  `MAXOCCURS` must be greater than or equal to **`MINOCCURS`**, but not zero.

- **`MAXBYTES`**

  This parameter specifies the maximum size of a field in bytes when the byte-length of the field instance can vary based on the number of elements in a dynamic array. **`MAXBYTES`** and **`BYTES`** are mutually exclusive.

  The value of **`MAXBYTES`** must be greater than or equal to the maximum total of the byte values of all fields nested under this field.

  The **`MAXBYTES`** parameter is required and valid only in the following cases:

  – The field is defined as a dynamic array.

  – For a field defined as a static array or for a structure that contains a nested field that is defined as a dynamic array.

- **`STARTAFTER`**

  This parameter specifies the name of the field that directly precedes this field in the segment. The name that is specified must be the value of the **`EXTERNALNAME`** parameter in the definition of the referenced field.

  `STARTAFTER` is required and valid only when the starting position of a field cannot be calculated. The starting position cannot be calculated because the field is preceded at a prior offset by a field that is defined as a dynamic array.

  The **`STARTAFTER`** parameter cannot be specified on fields that define an array field as a parent.

- **`RELSTART`**

  This parameter specifies the starting position of a field that is defined as an element of an array or a structure. Consider the following points:

  – For fields that specify an array field as a parent, **`RELSTART`** is required.

  – For fields that specify a structure as a parent, **`RELSTART`** is required only when a dynamic array precedes the structure at any prior offset in the segment.

`RELSTART` is the starting byte offset of the field relative to the start of the array or structure.

Changes to remarks for DBDs and PSBs

The following existing IMS macros are updated to allow up to 256 characters to be stored as a remark for a DBD or PSB. Users can specify comments that are stored in the DBDLIB, PSBLIB, ACBLIB, and IMS catalogs. This notation helps ensure that comments are not lost if the source is lost:

- PSB remarks

  PCB, SENFLD, and SENSEG

- DBD remarks

  DBD, SEGM, FIELD, XDFLD, LCHILD, DATASET, and AREA

5.4.2 Get Unique Record DL/I call

IMS V12 provides a new DL/I call specifically for the IMS catalog. The Get Unique Record (GUR) call retrieves the metadata for an IMS DBD or PSB from the catalog database. To read all the segments in this database record, the GUR call functions similarly to a Get Unique (GU) call followed by a series of Get Next within Parent (GNP) calls. This call can be used only to retrieve the metadata definition of an IMS DBD or PSB from the catalog database.

The GUR call reads an entire database record from the catalog database and returns an XML document that contains the metadata definition for the requested IMS resource (DBD or PSB). If the XML document is larger than the IOAREA that is provided by the application, subsequent GUR calls can be issued to retrieve the balance of the XML document. This process is done by passing back a token that was returned by the initial GUR call.

The data is buffered by IMS. This buffering is based on the assumption that the entire XML document is needed to minimize the processor usage when you ask for subsequent blocks of data to be returned. The advantage is that you do not need to overestimate the size of the IOAREA. If the IOAREA is too small, the call still succeeds and the balance of the data can be requested without having to reread the catalog database. This new call simplifies access to the database definition metadata and minimizes the processor usage when retrieving it.

If you pass a token with a value of zero, IMS assumes that it is a new GUR call and starts from the beginning.

DFSDDLT0 and IMS Restructured Extended Executor (REXX) were updated to add special processing for the IMS catalog and the XML that is returned from a GUR call.

The GUR call builds an entire XML instance document from the information in the catalog database. Internally, each GUR call with a zero AIBRTKN runs a GU and multiple GNP calls to read the IMS catalog. When the I/O area is too small for output, the output buffer is kept. A token is returned in the application interface block (AIB) to allow the application to resume copying data from the buffer to I/O area on subsequent calls.

GUR reads an entire record. SSAs can be coded starting with the HEADER and then the DBD or PSB. GUR does not function like GU calls in which you can read a segment with one qualified SSA, then get a lower level segment with a different SSA.

The GUR call can be used to access only an IMS catalog database, and the call requires the AIB interface. If an attempt is made to use the GUR call through the ASMTDLI, CBLTDLI, or PLITDLI interfaces, IMS returns a “DE” status code.

During initialization, IMS reads the time stamps from the ACBLIB and stores them into the database control blocks. The GUR call uses the time stamp to find the active member that is used by IMS (or the call finds the first record only) when IMS does not have a time stamp. IMS does not have a time stamp for resources that are not defined to this IMS.

Segment search arguments that are used with the GUR call

Typically, segment search arguments (SSAs) are needed when the GUR call is used to identify the database or PSB definition to be retrieved. SSAs are specified on the GUR call in the same way as other GU calls to IMS; that is, **`field <relational operator> field-value`**.

Example 5-14 shows an application program that uses the GUR call to retrieve the metadata for the S2U1DBD database.

Example 5-14 Sample application that uses the GUR DL/I call

```
Identification division.
         program-id. gur.
       Environment division.
       Data division.
        Working-storage section.
        01 dli-insert pic x(4) value 'ISRT'.
        01 dli-gur    pic x(4) value 'GUR'.
        01 dli-gu     pic x(4) value 'GU'.
        01 header-ssa.
           02 filler          pic x(8)  value 'HEADER  '.
           02 filler          pic x(1)  value '('.
           02 ssa-field-name  pic x(8)  value 'RHDRSEQ '.
           02 ssa-boolean     pic x(2)  value '= '.
           02 ssa-field-value pic x(16) value 'DBD     S2U1DBD '.
           02 filler          pic x(1)  value ')'.
         01  AIB.
           02 AIBRID             PIC x(8).
           02 AIBRLEN            PIC 9(9) USAGE BINARY.
           02 AIBRSFUNC          PIC x(8).
           02 AIBRSNM1           PIC x(8).
           02 AIBRSNM2           PIC x(8).
           02 AIBRESV1           PIC x(8).
           02 AIBOALEN           PIC 9(9) USAGE BINARY.
           02 AIBOAUSE           PIC 9(9) USAGE BINARY.
           02 AIBRESV2           PIC x(12).
           02 AIBRETRN           PIC 9(9) USAGE BINARY.
           02 AIBREASN           PIC 9(9) USAGE BINARY.
           02 AIBERRXT           PIC 9(9) USAGE BINARY.
           02 AIBRESA1           USAGE POINTER.
           02 AIBRESA2           USAGE POINTER.
           02 AIBRESA3           USAGE POINTER.
           02 AIBRESV4           PIC x(40).
           02 AIBRSAVE     OCCURS 18 TIMES USAGE POINTER.
           02 AIBRTOKN     OCCURS 6 TIMES  USAGE POINTER.
           02 AIBRTOKC           PIC x(16).
           02 AIBRTOKV           PIC x(16).
           02 AIBRTOKA     OCCURS 2 TIMES PIC 9(9) USAGE BINARY.
        01 gur-returned-data pic x(32000).
        Linkage section.
         1 io-pcb.
           3 filler         pic x(60).
```

```
 1 db-pcb.
   3 filler         pic x(10).
   3 status-code    pic x(2).
   3 filler         pic x(20).
Procedure division using io-pcb db-pcb.
   move "DFSAIB  "    to AIBRID.
   move length of aib to AIBRLEN.
   move length of gur-returned-data to AIBOALEN.
   move spaces        to AIBRSFUNC.
   MOVE "DFSCAT00"    TO AIBRSNM1.
   call 'AIBTDLI' using dli-gur aib
         gur-returned-data header-ssa.
   set address of db-pcb to aibresa1.
   display '|' status-code '|' db-pcb.
   display 'aib return is ' AIBRETRN .
   display 'aib reason is ' AIBREASN .
   display gur-returned-data.
   goback .
End program gur.
```

The first few lines of the data that is returned by the GUR call from the program (see Example 5-14 on page 159), are shown in Example 5-15. The first 56 characters in the IOAREA are not part of the XML document.

Example 5-15   Sample data that is returned by a GUR call

```
_%      ?>        > ?  >            /> /%?>   `     <ns2:dbd xmlns:ns2="http://www.ibm.com/ims/DBD" dbdName="S2U1DBD
" timestamp="1215015125765" version="02/10/1114.51" xmlSchemaVersion="1"><access dbType="HIDAM"><hidam datxexit="N" pass
word="N" osAccess="VSAM"><dataSetContainer><dataSet  ddname="S2U1DB" label="DSG1" searchA="0" scan="3"><block size="0"/>
<size size="0"/><frspc fspf="0" fbff="0"/></dataSet></dataSetContainer></hidam></access>&lt;segment imsName="CUSTROOT" name
="CUSTROOT" encoding="Cp1047"><hidam label="DSG1"><bytes maxBytes="76"/><rules insertionRule="L" deletionRule="L" replac
ementRule="L" insertionLocation="LAST"/><pointer physicalPointer="TWINBWD" lparnt="N" ctr="N" paired="N"/></hidam><field
 imsDatatype="C" imsName="CUSTNO" name="CUSTOMERNUMBER" seqType="U"><startPos>1</startPos><bytes>4</bytes><marshaller><t
ypeConverter>BINARY</typeConverter></marshaller><applicationDatatype datatype="BINARY"/></field><field imsDatatype="C" n
ame="FIRSTNAME"><startPos>5</startPos><bytes>10</bytes><marshaller encoding="Cp1047"><typeConverter>CHAR</typeConverter>
</marshaller><applicationDatatype datatype="CHAR"/></field><field imsDatatype="C" name="LASTNAME"><startPos>15</startPos
><bytes>20</bytes><marshaller encoding="Cp1047"><typeConverter>CHAR</typeConverter></marshaller><applicationDatatype dat
atype="CHAR"/></field><field imsDatatype="C" name="DATEOFBIRTH"><startPos>35</startPos><bytes>10</bytes><marshaller enco
ding="Cp1047"><typeConverter>CHAR</typeConverter></marshaller><applicationDatatype datatype="CHAR"/></field><field imsDa
tatype="C" name="HOUSENAME"><startPos>45</startPos><bytes>20</bytes><marshaller encoding="Cp1047"><typeConverter>CHAR</t
ypeConverter></marshaller><applicationDatatype datatype="CHAR"/></field><field imsDatatype="C" name="HOUSENUMBER"><start
```

A better way to display this XML information is through a browser. Example 5-16 shows the same information as displayed by Microsoft Internet Explorer.

Example 5-16   Sample data that is returned by a browser

```
<ns2:dbd xmlns:ns2="http://www.ibm.com/ims/DBD" dbdName="S2U1DBD" timestamp="1215015125765"
version="02/10/1114.51" xmlSchemaVersion="1">
<access dbType="HIDAM">
<hidam datxexit="N" password="N" osAccess="VSAM">
<dataSetContainer>
<dataSet ddname="S2U1DB" label="DSG1" searchA="0" scan="3">
<block size="0" />
<size size="0" />
<frspc fspf="0" fbff="0" />
```

```
</dataSet>
</dataSetContainer>
</hidam>
</access>
&lt;segment imsName="CUSTROOT" name="CUSTROOT" encoding="Cp1047">
<hidam label="DSG1">
<bytes maxBytes="76" />
<rules insertionRule="L" deletionRule="L" replacementRule="L" insertionLocation="LAST" />
<pointer physicalPointer="TWINBWD" lparnt="N" ctr="N" paired="N" />
</hidam>
<field imsDatatype="C" imsName="CUSTNO" name="CUSTOMERNUMBER" seqType="U">
<startPos>1</startPos>
<bytes>4</bytes>
<marshaller>
<typeConverter>BINARY</typeConverter>
</marshaller>
<applicationDatatype datatype="BINARY" />
</field>
<field imsDatatype="C" name="FIRSTNAME">
<startPos>5</startPos>
<bytes>10</bytes>
<marshaller encoding="Cp1047">
<typeConverter>CHAR</typeConverter>
</marshaller>
<applicationDatatype datatype="CHAR" />
</field>
```

The following XML schemas for the documents are returned as responses to this call and are included in the IMS.ADFSSMPL data sets:

- DFS3XDBD.xsd (for DBD records)
- DFS3XPSB.xsd (for PSB records)
- SSAs can be specified only up to the DBD or PSB level in the catalog database

No more than two SSAs can be specified; one for the root HEADER segment and one for the DBD or PSB segment

Example 5-17 shows the IMS Test Program (DFSDDLT0) statements that are needed to process a GUR call. The **DATA** statement is needed to provide sufficient IO-AREA for the XML document that contains the database definition to be returned to DSFSDDLT0.

Example 5-17 DFSDDLT0 statements for GUR

```
S 1 1 1 1 1    DFSCD000      AIB
L        GUR   HEADER  (RHDRSEQ ==DBD     S2U1DBD )
L  Z9999 DATA
```

5.4.3 IMS catalog access

When an IMS application program requires access to the metadata in the catalog, a PSB to access the catalog database is automatically attached to the PSB that is loaded for the application. IMS can then use that PSB to access the metadata in the IMS catalog. There is no need to specifically define the catalog database in your PSB.

Direct catalog interface

An application user can issue a DL/I call by using the PCB that directly references the IMS catalog DBD if they must access the metadata.

The application operates as a normal DL/I application. For example, it can use the new GUR DL/I call to read the entire record (root and all the child segments) from the IMS catalog.

Indirect catalog interface

The indirect catalog interface is used when a process needs access to the metadata but is not accessing the IMS catalog directly. The application processes as usual and can issue DL/I calls with a PSB that does not reference with the catalog DBD. When a process needs access to the metadata (in the past, the Java class from the DLIMODEL utility was used; however, you can now use the IMS Explorer for Development), IMS dynamically attaches a PSB with a catalog PCB, as shown in Figure 5-9.

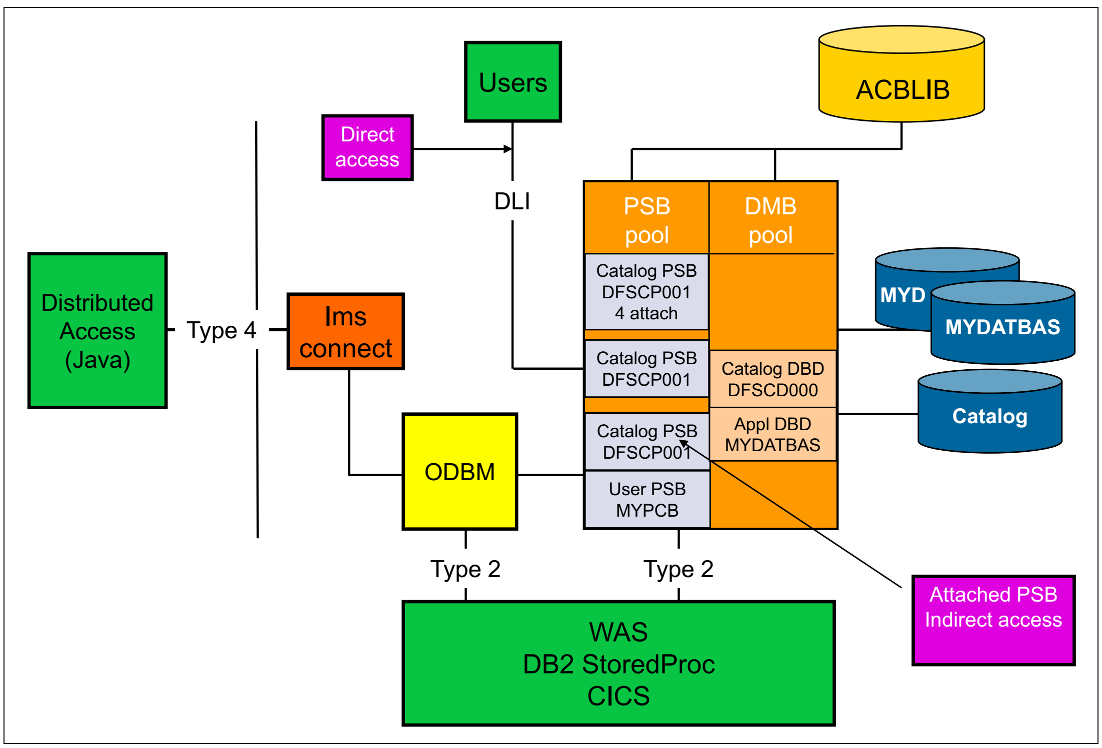


Figure 5-9 IMS catalog access

The catalog PCB is used by the DL/I engine without any external effects on the calling application. The normal mode of operation is that you are reading only the IMS catalog with a PROCOPT=G (read with integrity). The name of the main catalog PCB is DFSCAT00.

For message processing program (MPP) applications that are running in an online IMS, the dynamic attach of a PSB that contains the IMS catalog PCB occurs at the first reference; that is, during the first DL/I call. The database is accessed with deferred scheduling, so the scheduling occurs only when the first call is made that needs access to the IMS catalog. The IMS catalog database availability does not affect application availability. If the IMS catalog is offline, you can get an abend (U3303), but that does not cause the application to end.

In a DL/I batch environment, the dynamic attach of the IMS catalog PCB occurs during the batch initialization. Scheduling is done only once for the batch when the region starts. A batch job loads the IMS catalog PSB for later reference if the catalog is enabled in the DFSDFxxx PROCLIB member.

5.4.4 SSA enhancements

The new SSA format, *get by offset,* allows a new search function by offset and length, instead of field name. Fields are no longer required to be defined in the DBD. Consider the following points:

- Support is added for DFSDDLT0 and IMS REXX.
- Performance is the same as a non-key field search.
- IMS scans the database to look for matches.
- SSA contains offset and length followed by operator and value.

In Figure 5-10, an SSA search on the field is issued, although the type is not known by the DBD.

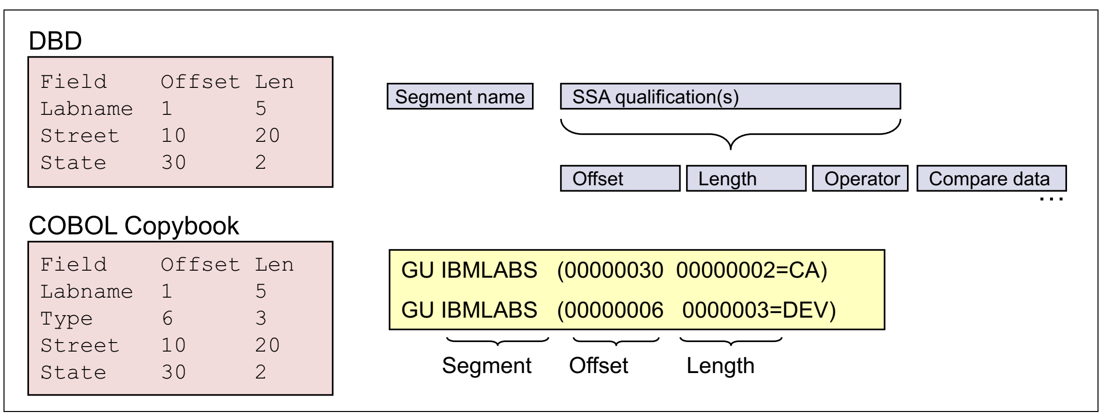


Figure 5-10 Using get by offset

Field level sensitivity is the mechanism with which the user can change the layout of the segment that is returned in the I/O area. The fields can be reordered, omitted, or spaces can be added between, as shown in Figure 5-11. The new SSA qualifications can be used with the existing SSA qualification format. For example, the following SSA combination is valid for the fields labname and type:

```
GU IBMLABS*O (LABNAME =SVL  &00000006 00000003=DEV)
```

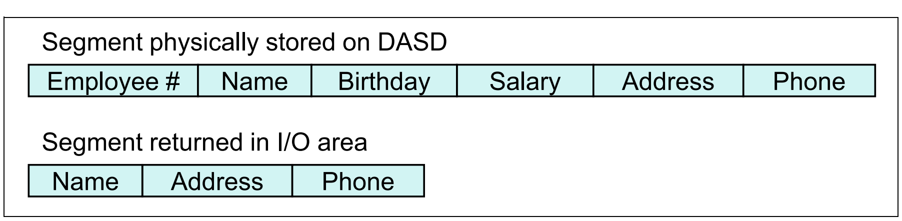


Figure 5-11 Field sensitivity

In this request, the SSA offset is relative to the physical starting position in the segment that is visible to your program. If you attempt to search for fields that are outside of the sensitive area (that is, past the end of the segment), IMS returns an AK status code. If a segment is not found that matches the SSA qualifications, a GE or GB status code is returned.

5.5 Enhancements to the IMS Universal drivers

All IMS Universal drivers are enhanced to use the IMS catalog and provide the following features:

- Direct access to IMS metadata in the catalog is available.
- Drivers are no longer require the separate Java metadata class.
- Drivers are no longer file system-dependent for metadata (there is virtual deployment support).
- The metadata is trusted and up-to-date.
- The Application Development community can access any IMS database that is defined in the catalog.
- There is new complex and flexible data-type support.

5.5.1 Access to the IMS databases from Java

You must establish a connection before any data can be accessed from DL/I databases. Traditionally, access with IMS and DLI requires a PSB, which is also true for Java. Traditional languages, such as COBOL and PL/I, work with a segment content, which is accessed by the SSA protocol. Java supports the following methods to access DL/I databases through the Universal drivers (type-4 for distributed and type-2 for local):

- SSA oriented (for DL/I clients)
- JDBC clients

JDBC requires the support of metadata for the accessed information. In previous releases, this support was provided by DatabaseView classes that are generated by the DLIModel utility. Now you must use IMS Explorer for Development.

IMS V12 added a more centralized approach by putting the metadata in to a catalog. The usage of metadata information offers many access perspectives to the Java language (case mapping, structure selects, and so on) that are not available in other languages.

The metadata support can come from the DLIModel or the IMS catalog. This choice must be made at connection time, as shown in Figure 5-12.

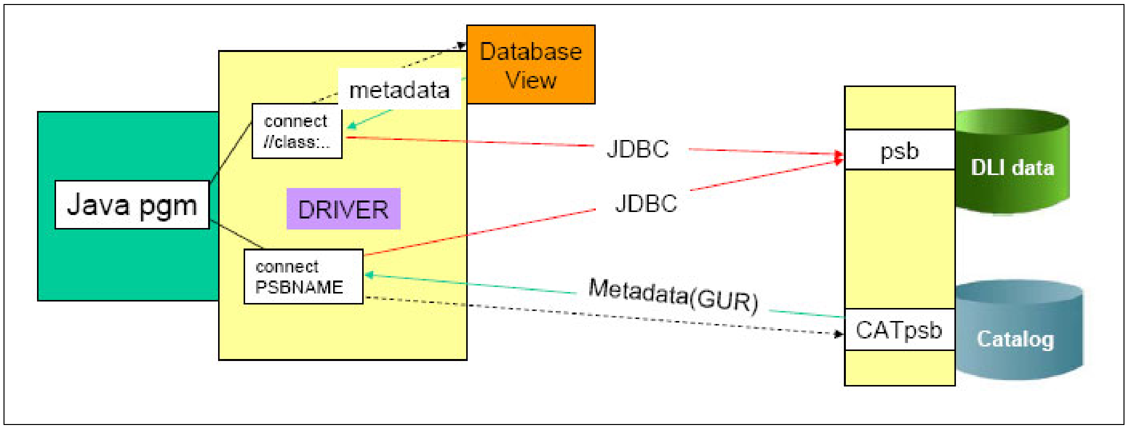


Figure 5-12 Connection of a Java program through the drivers

The following elements are required before you can access the DL/I databases:

- A PSB (for language Java) must exist.
- A DatabaseView class or an IMS catalog for obtaining metadata:
  - A DatabaseView class must be generated by the IMS Explorer for Development.
  - Metadata information is available in the catalog.

Connecting to IMS for use as a JDBC client

Example 5-18 shows a code excerpt of how to obtain a type-4 connection (with an IP address and port number) with IMS for a JDBC client.

Example 5-18 Obtain a connection with IMSDataSource for use with JDBC

```
JDBC connection:

import java.sql.Connection;
import java.sql.SQLException;
import com.ibm.ims.jdbc.IMSDataSource;
...
      IMSDataSource ds = new IMSDataSource();
      Connection conn = null;

      ds.setDatastoreName(alias);// IMS alias name defined in ODBM
      ds.setDatabaseName(psbName);
      // ds.setMetadataURL(url);---> deprecated

      ds.setUser(user);
      ds.setPassword(password);
      ds.setDriverType(driverType);

      // optional settings
      ds.setLoginTimeout(seconds);
      ds.setDescription(description);
      try {
         ds.setLogWriter(out);// set to a java.io.PrintWriter object
      } catch (SQLException e) {
         // handle exception
```

```
      }

      // JDBC type-4 driver specific settings
      if (driverType == IMSDataSource.DRIVER_TYPE_4) {
         ds.setSSLConnection(enableSSL);
         ds.setDatastoreServer(host);// IP address or DNS name of the
                        // LPAR where ICON resides
         ds.setPortNumber(port);// ICON DRDA port number
      }

      try {
         // Establish a connection using the data source
         conn = ds.getConnection();

         // do some work here

         conn.close();
      } catch (SQLException e) {
         // handle exception
      }
   }
```

Example 5-18 on page 165 shows the usage of IMSDatasource, which gets its properties through several setters and the **getConnection()** method to obtain the connection object.

As shown in Example 5-19, the **setMetadataURL()** method on the IMSDataSource is now deprecated and replaced by **setDatabaseName(url)**.

Example 5-19 Obtain a connection with PSB for use with DL/I

```
DLI connection:

import com.ibm.ims.dli.DLIException;
import com.ibm.ims.dli.IMSConnectionSpec;
import com.ibm.ims.dli.IMSConnectionSpecFactory;
import com.ibm.ims.dli.PCB;
import com.ibm.ims.dli.PSB;
import com.ibm.ims.dli.PSBFactory;
import com.ibm.ims.jdbc.IMSDataSource;
...
      IMSConnectionSpec connSpec
         = IMSConnectionSpecFactory.createIMSConnectionSpec();
      PSB psb;
      PCB pcb;

      connSpec.setDatastoreName(alias);
      connSpec.setDatabaseName(databaseName);
      // connSpec.setMetadataURL(url);---> deprecated

      connSpec.setUser(user);
      connSpec.setPassword(password);
      connSpec.setDriverType(driverType);

      if (driverType == IMSDataSource.DRIVER_TYPE_4) {
         connSpec.setSSLConnection(enableSSL);
         connSpec.setDatastoreServer(host);
```

```
      connSpec.setPortNumber(port);
   }

   try {
      psb = PSBFactory.createPSB(connSpec);
      pcb = psb.getPCB(pcbName);

      // do some work here

      psb.commit();// or alternatively, psb.rollback()

      psb.close();
      pcb.close();
   } catch (DLIException e) {
      // handle exception
   }
```

DL/I connection: For a DL/I client, the PCB is the target for other interactions.

The **`setDatabaseName(url)`** method sets the name of the target IMS databases to be accessed.

If the metadata repository is the IMS catalog, the database name (URL) is the name of the PSB that contains the databases to be accessed.

If the metadata repository is the DatabaseView class, the database name is the fully qualified name of the Java metadata class that is generated by the IMS Explorer for Development. In this case, the name must begin with `class://`.

JDBC client connection: For a JDBC client, the Connection object is the target for all other interaction with the DL/I databases.

Connecting to IMS for use as a DL/I client

Example 5-19 on page 166 also shows an excerpt of the code that is required to obtain access to the DL/I data from a DL/I client through the PCB object. Through a DL/I client, access takes place as in the other languages, with PCB, SSAs, and function codes.

5.5.2 Using the metadata information in the DL/I access

With traditional languages (COBOL, PL/I, C, and so on), reading a segment brings a complete record in to an I/O area known as a *byte area*. This area must be interpreted. The byte area contains value fields with structures and substructures. In these languages, the extract of the elementary values, which can be in different formats (char, binary, or packed decimal), is straightforward. The extract is done by overlaying the I/O area with redefinitions (COBOL copybooks and PL/I or C include members). This technique is not available for Java.

The changes to the DBDGEN macros are described in 5.4.1, “DBD and PSB source changes” on page 152. The changes on DBDGEN, and the extensions to the Java access classes, can be used for an enhanced access approach of the DL/I databases by Java programs. The changes in DBDGEN provide for the extension of the metadata in the DBD and to store it in the catalog. Other metadata information that is implicitly present in the redefinition sections also can be captured into the catalog.

By using the metadata, Java (JDBC and DL/I) offers powerful detailed and controlled access to the information in the databases. When you program a Java client to access the DL/I databases, you can use the following different styles:

- JDBC SQL approach, with or without IMS extensions
- IMS extended DL/I style

Struct and Array objects: Traditionally, when a Struct or an Array object is created, you must define it in a bottom-up manner that starts with the Struct attributes or the Array Elements.

IMS can be used for the top-down creation of a Struct or an Array object. You can define the Struct or Array object and then set the attributes and elements. It avoids potential complexity when you deal with nested structures. It also allows for setting and getting Struct and Array attributes by the attribute name.

Along with the metadata support for Java, several enhancements are available for Java. These enhancements are described in the following sections.

Variable length segment support

The IMS Universal drivers for both type-2 and type-4 database access are enhanced to support variable length database segments. Support is added to the Universal drivers by the following maintenance:

- Version 11: APAR PM14766
- Version 12: APAR PM25951

Because the standards for JDBC do not have a concept of segment length, the default behavior is that the JDBC driver manages variable length segments internally. The behavior is managed for read, update, and insert operations, without any additional action from the client.

SQL clients can directly read and update the length field of a variable length segment by explicitly requesting access when the connection is created.

DL/I clients always have access to the length field.

Variable length segments contain a 2-byte length (LL) field that identifies the size of the segment instance. The Universal drivers are now sensitive to the LL field of a variable length segment. The drivers manage the I/O area of the segment instance on standard *create, retrieve*, *update*, and *delete* calls. A new Connection Property *llField* determines whether the LL field of the segment is displayed to the program or not.

The New Connection Property llField can be set to the following statuses:

- `llField = true`

  The LL field of variable length segments is displayed to the user in the SQL and DL/I interfaces. The user fully manages the LL field for the segment instance. The LL field is how users manage variable length segments.

- `llField = false` (default)

  The IMS Universal driver automatically manages the LL field for the user.

> **Variable length support:** Variable length support is introduced for DLIModel and the IMS catalog metadata.

Example 5-20 shows the variable length information that related to the segment WARD in the DatabaseView.

Example 5-20 Variable segment length information in DatabaseView class

```
   static DLISegment PHDAMVARWARDSegment = new DLISegment
         ("WARD","WARD",PHDAMVARWARDArray,32,900,DBType,PHAM,false);
//                                 Minlength, Maxlength
```

Example 5-21 shows the variable length information in the catalog.

Example 5-21 Variable segment length information in IMS catalog: XML

```
   &lt;segment imsName="WARD" name="WARD">
      <phdam>
         <bytes minBytes="32" maxBytes="900"/> <======minbytes,maxbytes
      </phdam>
   ....
   &lt;/segment&gt;
```

Structure support

Structures are data structures that store a combination of values. Structure support was added specifically for the IMS catalog in IMS V12 to represent common application metadata. Example 5-22 represents a simple structure with the name ADDRESS-INFO. In Java, it is an object.

Example 5-22 COBOL structure example

```
01 PERSON.
   02 PERSNR     PIC X(6).
   02 ADDRESS-INFO.
      04 CITY    PIC X(15).
      04 STREET  PIC X(25).
      04 ZIP     PIC X(5).
```

Structure metadata is stored in the IMS catalog. IMS Universal drivers can retrieve the metadata on Structs as XML through the GUR call, as shown in Example 5-23.

Example 5-23 Structure information in the IMS catalog

```
..........
<field name="ADDRESS_INFO">
   <startPos>6</startPos>
   <bytes>45</bytes>
   <marshaller encoding="CP1047">
      <typeConverter>STRUCT</typeConverter>
   </marshaller>
   <applicationDatatype datatype="STRUCT"/>
   <field imsDatatype="C" name="CITY">
```

```
      <startPos>1</startPos>
      <bytes>15</bytes>
      <marshaller encoding="CP1047">
         <typeConverter>CHAR</typeConverter>
      </marshaller>
      <applicationDatatype datatype="CHAR"/>
   </field>
   .........other fields
</field>
```

Example 5-24 shows the code for the JDBC SQL retrieve of the structure (PERSON_INFO). See the COBOL definition in Example 5-22 on page 169. In the code, you extract the subfield values from the structure object. Two approaches are shown in Example 5-24.

Example 5-24 JDBC SQL for structure retrieve

```
   Connection conn = ....
   Statement st = conn.createStatement("SELECT * FROM”
                     + pcbName +".PERSON WHERE PERSNR = ‘XXXXXX?”);
   ResultSet rs = st.executeQuery();
   rs.next();
// 2 ways to retrieve the information
//----------------------------------------------------------------------
// 1) standard SQL
   Struct addressInfo = (Struct)rs.getObject("ADDRESS_INFO");
   Object[] addressInfoAttributes = addressInfo.getAttributes();
   String city = (String) (addressInfoAttributes[0]).trim());
   String street = (String) (addressInfoAttributes[1]).trim());
   String zip = (String) (addressInfoAttributes[2]).trim());
//----------------------------------------------------------------------
// 2) with IBM/IMS extensions
   StructImpl addressInfoimpl = (StructImpl)rs.getObject("ADDRESS_INFO");
   String city = addressInfoimpl.getString(“CITY”);
   String street = addressInfoimpl.getString(“STREET”);
   String zip = addressInfoimpl.getString(“ZIP”);
//----------------------------------------------------------------------
```

With the StructImpl class, you have direct access to the structure fields with the adequate getters.

Next, new values are provided for the fields, and a JDBC/SQL update stores the new structure in the DL/I database, as shown in Example 5-25.

Example 5-25 Updating the PERSON segment with JDBC

```
   Connection conn = ....
   PreparedStatement ps = conn.prepareStatement("UPDATE " + pcbName + ".PERSON
                  SET ADDRESS_INFO =? WHERE PERSNR = “XXXXXX");
//-------------------------------------------------------------------
// 1) standard SQL
   Object[] newAddressInfoAttribute = new Object[]
                         {“LEUVEN”,”BONDGENOTENLAAN”, “3000”};
   Struct newAddressInfoStruct = conn.createStruct(pcbName +
                         ".PERSON.ADDRESS_INFO", newAddressInfoAttribute);
   ps.setObject(1, newAddressInfoStruct);
//-------------------------------------------------------------------
// 2) with IBM/IMS extensions
   StructImpl newaddressInfoImpl =
           (StructImpl)conn.createStruct(pcbName + ".PERSON.ADDRESS_INFO");
   newaddressInfoImpl.setString(“CITY”,”LEUVEN”);
   newaddressInfoImpl.setString(“STREET”,”BONDGENOTENLAAN”);
   newaddressInfoImpl.setString(“ZIP”,”3000”);
   ps.setObject(1, newaddressInfoimpl);
   //-------------------------------------------------------------------
   int numberOfSuccessfulUpdates = ps.executeUpdate();
```

The scenarios that are used in Example 5-25 showed code for a JDBC client. Example 5-26 shows a Java code excerpt for a retrieve with a DL/I client.

Example 5-26 DL/I access for structure retrieve

```
   PCB pcb = ... {
   SSAList ssaList = pcb.getSSAList("PERSON");
//specify a qualified SSAList for segment PERSON = XXXXXX field PERSNR
   ssaList.addInitialQualification("PERSON","PERSNR",SSAList.EQUALS, "XXXXXX");
//Retrieve all fields from PERSON (fixed segment)
   ssaList.markAllFieldsForRetrieval("PERSON", true);
//Creates I/O area for data
   Path path = ssaList.getPathForRetrieve();
//retrieve the data
   pcb.getUnique(path, ssaList, true);
   DBStruct personinfo = (DBStruct)path.getObject("PERSON_INFO");
//----------------------------------------------------------------------
// 1)
   Object[] addressInfoAttributes = personinfo.getAttributes();
   String city = ((String)addressInfoAttributes[0]).trim();
   String street = ((String) addressInfoAttributes[1]).trim();
   String zip = ((String) addressInfoAttributes[2]).trim();
//----------------------------------------------------------------------
// 2)
   String city = personinfo.getString(“CITY”).trim();
   String street = personinfo.getString(“STREET”).trim();
   String zip = personinfo.getString(“ZIP”).trim();
```

Arrays

If the correct metadata is assembled in the catalog, *arrays* are also supported. As shown in Example 5-27, arrays are data structures that store a repeating combination of values.

Example 5-27 COBOL array

```
01 STUDENT.
   02 STUDENTNAME PIC X(25).
   02 AGE PIC 9(2) COMP.
   02 COURSE OCCURS 5 TIMES.
      04 COURSENAME PIC X(15).
      04 INSTRUCTOR PIC X(25).
      04 COURSEID PIC X(5).
```

Dynamic and static array support was added specifically for the IMS catalog in IMS 12 to represent common application metadata. IMS Universal drivers support only static arrays. The drivers can retrieve the metadata on arrays as XML through the GUR call.

A proprietary method of setting the elements within a DBArrayElementSet object is added. The method is similar to a ResultSet. A specific element can be positioned in an array with the following methods:

- **`next()`**
- **`previous()`**
- **`first()`**
- **`last()`**
- **`absolute(int index)`**

Also, the following common getters and setters are provided for nested fields within an array element:

- setBoolean(String, Boolean)
- setString(String, String)
- getBoolean(String)
- getString(String)

Example 5-28 shows the retrieve of the STUDENT segment, which contains an array. The DBArrayElementSet class was used here.

Example 5-28 Retrieve of the segment STUDENT

```
   String[] coursename =new String[5];
   String[] instname =new String[5];
   String[] courseid =new String[5];
   st = conn.createStatement();
   rs = st.executeQuery("SELECT * FROM " + pcbName +
                  ".STUDENT WHERE STUDENTNAME = XXXX" );
   rs.next()
   String studname = rs.getString("STUDENTNAME");
   short age = rs.getShort("AGE");
// 2 ways to retrieve the information
//--------------------------------------------------------------------
// 1) standard SQL
   Array fivecourses = rs.getArray("COURSE");
   Struct[] fivecourseselem = (Struct[]) fivecourses.getArray();
   // Each array element is represented as a Struct
   for (int arrayIdx = 0; arrayIdx < fivecourseselem.length; arrayIdx++) {
      Object[] courseinfo = fivecourseselem[arrayIdx].getAttributes();
```

```
      coursename[arrayIdx] = (String) courseinfo[0];
      instname[arrayIdx] = (String) courseinfo[1];
      courseid[arrayIdx] = (String) courseinfo[2];
   }
//---------------------------------------------------------------------
// 2) with IBM/IMS extensions
   int arrayIdx = 0;
   ArrayImpl fivecoursesImpl = (ArrayImpl)rs.getArray("COURSE");
   DBArrayElementSet fivecoursesArrayElementSet = fivecoursesImpl.getElements();
   try {
      while (fivecoursesArrayElementSet.next()); {
         coursename[arrayIdx] =
                   fivecoursesArrayElementSet.getString("COURSENAME");
         String instname[arrayIdx] =
                   fivecoursesArrayElementSet.getString("INSTRUCTOR");
         String courseid[arrayIdx] =
                   fivecoursesArrayElementSet.getString("COURSEID");
         arrayIdx++;
      }
   }  catch (Exception e {
      .....
   }
//---------------------------------------------------------------------
```

Example 5-29 shows the insertion of a new STUDENT segment. The two ways of JDBC coding are shown again.

Example 5-29 Insertion of a new STUDENT segment

```
   PreparedStatement ps = conn.prepareStatement
         ("INSERT INTO " + pcbName + ".STUDENT ("STUDENTNAME", "AGE", "COURSE")
         VALUES( ? ? ?);
   String coursenm[] = {“HISTORY”,”ENGLISH”,”FRENCH”,”ITALIAN”,”JAVA”};
   String teacher[] = {“SMITH”,”OBAMA”,”SARKOZY”,”PAOLO”,”SVLTEACHER”};
   String courseid[] = {“10”,”51”,”52”,”55”,”40”};
   ps.setString(1,”FREDERIK”);
   ps.setShort(2,41);
//---------------------------------------------------------------------
// 1) standard SQL
// fill now the ARRAY object
   Struct[] courseArrayElements = new Struct[5];
   for (int arrayIdx = 0; arrayIdx < 5; arrayIdx++) {
      Object[] courseAttributes = new Object[]
            {coursenm[arrayIdx], teacher[arrayIdx], courseid[arrayIdx]};
      courseArrayElements[arrayIdx] = conn.createStruct
            (“COURSE", courseAttributes);
   }
   Array courseArray = conn.createArrayOf(“COURSE, courseArrayElements);
   ps.setArray(3,courseArray);
//---------------------------------------------------------------------
// 2) with IBM/IMS extensions
   int arrayIdx = 0;
   ArrayImpl fivecoursesImpl = ((ArrayImpl) ((ConnectionImpl)conn).createArrayOf
            (pcbName + ".STUDENT.COURSE"));
   DBArrayElementSet fivecoursesArrayElementSet = fivecoursesImpl.getElements();
   try {
```

```
      while (fivecoursesArrayElementSet.next()); {
         fivecoursesArrayElementSet.setString(“COURSENAME”,coursenm[arrayIdx]);
         fivecoursesArrayElementSet.setString(“INSTRUCTOR”,teacher[arrayIdx]);
         fivecoursesArrayElementSet.setString(“COURSEID”,courseid[arrayIdx]);
         arrayIdx++;
      }
   } catch (Exception e) {
   }
   ps.setArray(3,fivecoursesImpl);
//-----------------------------------------------------------------------
   int numberOfSuccessfulInserts = ps.executeUpdate();
```

Mapping support

A map is metadata that describes how a field (or set of fields) is mapped for a particular segment instance. Metadata captures the various cases, and for each case, defines the set of fields to be used for that case. Maps can be defined to the catalog.

Maps are interpreted at run time by the IMS Universal drivers and the applicable data elements are returned based on the runtime case of the segment instance. Figure 5-13 shows a case selection that is based on the control field Policy Type.

Policy Type	Property Type	Rooms	MValue	Address	Make	Model	Year	HValue	Color
M	-	-	-	-	Ford	Escort	1989	2K	Red
H	Single Family	5	500K	555 Disk Drive Way, 95141	-	-	-	-	-

Figure 5-13 Insurance segment mapped multiple ways based on the Policy Type control field

All case fields are displayed in the metadata. There is no concept of maps or cases at the SQL level. The support by metadata creates a requirement that all case fields have unique names in a table.

Figure 5-13 shows the SQLview of the data. Although this representation gives the impression that much of the space is empty on the disk, the reality is different. All maps must be the same length, and in this case, the physical view of the segment is similar to what is shown in Figure 5-14.

Policy Type	Make	Model	Year	HValue	Color
M	Ford	Escort	1989	2K	Red

	Property Type	Rooms	MValue	Address
H	Single Family	5	500K	555 Disk Drive Way, 95141

Figure 5-14 Segment view on disk

Figure 5-14 also shows that the control field Policy Type is not really a part of the mapping.

Example 5-30 shows an excerpt of the case mapping information in the metadata.

Example 5-30 IMS Universal drivers pull metadata from the IMS catalog GUR call as XML

```
<field imsDatatype="C" imsName="POLTYPE" name="PolicyType"> <-CASE Control field
   <startPos>1</startPos>
   <bytes>1</bytes>
   <marshaller encoding="CP1047">
      <typeConverter>CHAR</typeConverter>
   </marshaller>
   <applicationDatatype datatype="CHAR"/>
</field>
<mapping dependingOnField="PolicyType">
   <case name="HOUSE"> ...............................<- CASE
      <dependingOnFieldValue valueDatatype="C" value="H"/>
      <field imsDatatype="C" imsName="PROPTYPE" name="PropertyType">
         <startPos>2</startPos>
         <bytes>15</bytes>
         <marshaller encoding="CP1047">
            <typeConverter>CHAR</typeConverter>
         </marshaller>
         <applicationDatatype datatype="CHAR"/>
      </field>
      ...
   </case>
   ...
</mapping>
```

Maps and cases are a new feature that are added to the SQL and DL/I interfaces and are functionally identical. The use of case-mapping support offers the following specific create, retrieve, update, and delete behaviors:

- SQL Select and DL/I Read

  When a case is not the active case that is based on the control field of the map, its fields are treated as null fields.

- SQL Insert and DL/I Create

  You cannot insert values for the fields of a case unless the insert also includes the value for the control field that makes the case active.

- SQL Update and DL/I Update

  You cannot update the values for the fields of a case unless the case is active, or you can update the values if the control field is also updated to a value that makes the case active.

- SQL Delete and DL/I Delete

  No new behavior is observed.

Example 5-31 shows a case and mapping retrieve.

Example 5-31 Cases and mapping

```
   st = conn.createStatement();
   rs = st.executeQuery("SELECT * FROM " + pcbName + ".INSURANCES");
   while (rs.next())  {
      byte policytype = rs.getByte("PolicyType");
      switch (policytype) {
         case ('M'):
```

```
            String make = rs.getString("Make");
            String model1 = rs.getString("Model");
            short year = rs.getShort("Year");
            int mvalue = rs.getInt("MValue");
            String color = rs.getString("Color");
         case ('H'):
            String propertyType = rs.getString("Make");
            short room = rs.getShort("Room");
            int hvalue = rs.getInt("HValue");
            String address = rs.getString("Address");
      }
   }
```

Redefines

Redefines are overlapping fields. Redefines are a way of remapping a field in the database. A COBOL structure example of a redefine is shown in Example 5-32.

Example 5-32 COBOL structure example

```
01 PERSON.
   02 ADDRESS PIC X(45).
   02 ADDRESS-INFO REDEFINES ADDRESS. <-------------------
      04 CITY PIC X(15).
      04 STREET PIC X(25).
      04 ZIP PIC X(5).
```

Example 5-33 shows the way this redefine information is stored in the catalog.

Example 5-33 Redefine metadata information in the IMS catalog

```
<field imsDatatype="C" name="ADDRESS">
   <startPos>1</startPos>
   <bytes>45</bytes>
   <marshaller encoding="CP1047">
      <typeConverter>CHAR</typeConverter>
   </marshaller>
   <applicationDatatype datatype="CHAR"/>
</field>

<field name="ADDRESS_INFO" redefines="ADDRESS> .............< redefine
   <startPos>1</startPos>
   <bytes>45</bytes>
   <marshaller encoding="CP1047">
      <typeConverter>STRUCT</typeConverter>
   </marshaller>
   <applicationDatatype datatype="STRUCT"/>
   ...
</field>
```

Fields that overlap can be interchangeable. The search performance of a query depends on the following type of field in the qualification statement:

- Key fields: Fastest
- Searchable fields
- Not searchable (that is, application defined fields)

The universal drivers promote a field upward when queries are issued against IMS.

The following example shows how the key field, KEY, and the non-key field, NONKEY, redefine each other:

```
SELECT * FROM TBL WHERE NONKEY=A ===>becomes
SELECT * FROM TBL WHERE KEY=A
```

The trace and log files show the promotion in the SSA list that is sent from the IMS Universal driver to IMS.

Public converter interfaces

Public interfaces are added to the internal type converters of the IMS Universal drivers for clients to use in implementing their own type converter routines. Support is added in IMS V10 and later.

By using the new ConverterFactory class, users can create the following converter interfaces:

- DoubleConverter
- FloatConverter
- IntegerConverter
- LongConverter
- PackedDecimalConverter
- ShortConverter
- StringConverter
- UByteConverter
- UIntegerConverter
- ULongConverter
- UShortConverter
- ZonedDecimalConverter

The converter classes contain a *getter and setter method* for converting the data type to a binary representation.

The IMS catalog is built to store information about the fields with a user-defined type. By using this defined type, users can interpret binary data that is stored in IMS in whatever form is required for their application.

Example 5-34 shows the IMS catalog metadata that contains a user-defined type.

Example 5-34 Field with a fully defined converter class

```
<field name="PACKEDDATEFIELD">
   <startPos>40</startPos>
   <bytes>5</bytes>
   <marshaller encoding="">
      <userTypeConverter>class://com.ims.PackedDateConverter</userTypeConverter>
      <property name="pattern" value="yyyyMMdd"/>
      <property name="isSigned" value="N"/>
   </marshaller>
   <applicationDatatype datatype="OTHER"/> <===== set to Other ====
</field>
```

User Type converters must extend the `com.ibm.ims.dli.types.BaseTypeConverter` abstract class.

Universal drivers include a user-defined type converter with its source for the PackedDate type, which stores date information as a PackedDecimal field, as shown in Example 5-35.

Example 5-35 User-defined type converter for the PackedDate type

```
public class PackedDateConverter extends BaseTypeConverter {

   public Object readObject(byte[] ioArea,int start,int length,Class objectType,
         Collection<String> warningStrings) throws ConversionException {
      ...
   }

   public void writeObject(byte[] ioArea, int start, int length, Object object,
         Collection<String> warningStrings) throws ConversionException {
      ...
   }
}
```

Application-transparent metadata access

Users can explicitly issue a GUR call through the following method, which is available only for DL/I clients:

```
PCB.getCatalogMetadataAsXML(String resourceName, byte[] resourceType)
```

The following input parameters are featured for this method:

- `resourceName`

  The name of the PSB or DBD resource to be retrieved.
- `resourceType` in the format PCB.PSB_RESOURCE or PCB.DBD_RESOURCE

  Identifies whether a PSB or a DBD resource is retrieved.

The call returns an XML document that contains the metadata for the resources that are requested. The XML document conforms to the IMS managed schemas. The following schemas are used when XML information is provided from the IMS catalog:

- DFS3XDBD.xsd
- DFS3XPSB.xsd

The source of the XML Schema Definitions (XSDs) is provided in the IMS sample library (SDFSSMPL).

Example 5-36 shows the explicit GUR call.

Example 5-36 Example of an explicit GUR call

```
byte[] gurOutput = pcb.getCatalogMetadataAsXML("STLIVP1", PCB.PSB_RESOURCE);
```

Example 5-37 shows the getCatalogMetaDataAsXML public method definition.

Example 5-37 Example of a getCatalogMetaDataAsXML public method definition

```
    /**
     * This method returns a byte array containing the requested catalog resource
     * as an XML document.
     * <p>The following code fragment illustrates how to retrieve the timestamp
     * (TSVERS) value from the IMS Catalog.
     * <blockquote>
     * <pre>
     * PCB pcb = psb.getPCB("DFSCAT00");
     * SSAList ssaList = pcb.getSSAList("HEADER", "DBD");
     * Path path = ssaList.getPathForRetrieveReplace();
     * pcb.getUnique(path, ssaList, false);
     * String timestamp = path.getString("TSVERS");
     * </pre>
     * </blockquote>
     *
     * @param resourceName the name of the PSB or DBD resource in the catalog
     * @param resourceType the type of resource (PCB.PSB_RESOURCE or
PCB.DBD_RESOURCE)
     * @param timestamp the TSVERS version for the resource following the
     *        pattern yyDDDHHmmssff
     * @return the resource metadata in XML
     * @throws DLIException if the resource was not found in the catalog or an
error
     *         occurs during processing
     * @see #PSB_RESOURCE
     * @see #DBD_RESOURCE
     */
    public byte[] getCatalogMetaDataAsXML(String resourceName,
                  byte[] resourceType, String timestamp)
                  throws DLIException
;
```

6

Database versioning

In this chapter, we describe the DB versioning feature of IMS 13 with which changes to database segment structures can be made while existing programs can continue to work without any changes or having to be recompiled.

We also describe the process of making segment length changes to provide new fields without forcing the recompilation of existing programs that do not need access to those new fields.

We separated the topic of DB versioning from the general Database description in Chapter 4, “Database and Database Recovery Control enhancements” on page 101because of its importance and its prerequisite (the IMS catalog), which is described in Chapter 5, “IMS catalog” on page 135.

This chapter includes the following topics:

- Overview of database versioning
- How versioning works
- Using database versioning

6.1 Overview of database versioning

Database versioning enables structural changes to a database while providing multiple views of the physical IMS data to application programs. Applications that are referencing a new physical database structure can be made without affecting applications that use the previous database structures. The changes that are enabled in this version of IMS are the ability to extend the length of a segment or segments without having to change all the existing programs that access those segments. This configuration enables the segment to be extended to include new fields that can be used by new programs or existing programs that are modified to use these new fields. Changing existing fields or adding fields before any existing fields in the segment is not supported.

These changes are normally implemented by recording the database description (DBD) source and running the DBD, PSB, and ACBGEN utilities. The user then unloads or reloads the database or uses the IMS 13 HALDB Alter function followed by performing an Online Change (OLC).

Application programs that do not require sensitivity to the new fields do not need to be modified. They can continue to access the database as they always did, which reduces the number of application changes, testing, and implementation tasks.

Database versioning support is for Full Function, HALDB, and DEDB database customers who need support for multiple views of the physical data to the following application needs:

- Implementing application changes over time.
- Ability to use application programs (for which there is no source code) after database structure changes.
- To avoid changing and testing large numbers of application programs that do not need access to the new structure.
- Applications that do not require sensitivity to the new physical database structure can continue to access the database without any modifications or recompilation.

Versioning support enables users to assign user-defined version IDs to different versions of the structure of a database. The user-defined version IDs are stored in the record for the database in the IMS Catalog. When the application requests access to the database, the application specifies the version of the database that it needs. If a version is not specified, by default the applications receive the version of the database structure at level set as the default level in the IMS system.

6.2 How versioning works

The objective of database versioning allows programs that have a defined view of a segment within a database to continue to use that view of the segment, even if the physical segment was extended to add more fields. The segments view of a segment is normally defined by a COBOL or PL/I copybook. IMS always retrieves the entire segment and passes the entire segment length into the IOAREA that is provided by the program in the DLI call. Therefore, for programs to continue to operate without changing the definition of the IOAREA, database versioning is needed.

IMS uses the IMS catalog to store the definition of the segments at a specific version of the database. The original version is called the *base version*. The most recent view (called the *CURRent view*) is a complete view of how the segment is physically defined. When a program requires a version of the database that is not the current version, IMS goes to the catalog to get the information for this version of the database. IMS compares the catalogs information with the current DBD and determine the differences. IMS then passes the requested version of the segment to the program in the IOAREA and not the complete segment from the buffer.

For example, take a customer database that features several segments types. Because we use the case where most of the original programs and PSB are not to be modified, the IMS system default of DBLEVEL=BASE is used.

The CUSTOMER segment copybook has six fields with a total length of 94 bytes, as shown in Figure 6-1. There was no more space on the segment for more fields.

```
01 CUSTOMER
   05 CUST-NO        PIC(06)
   05 FIRST_NAME     PIC(30)
   05 CUST_NAME      PIC(30)
   05 BIRTHDAY       PIC(10)
   05 PHONE          PIC(12)
   05 SEX            PIC(6)
```

Figure 6-1 CUSTOMER segment before versioning

The DBD is shown in Figure 6-2 on page 184 with all the fields defined and the external names are included as this DBD is used to populate the IMS catalog. The FIELD statements for the non-key fields are not required. The EXTERNALNAME parameters are also optional for the use of versioning.

```
      DBD NAME=G2CSTMRP,ACCESS=(PHIDAM,VSAM)
*
      SEGM    NAME=CUSTOMER,BYTES=94,PTR=TB
      FIELD   NAME=(CUSTNO,SEQ,U),START=1,BYTES=6,TYPE=C,
              EXTERNALNAME=CUST-NO
      FIELD   NAME=(FRSTNAME),START=7,BYTES=30,TYPE=C,
              EXTERNALNAME=FIRST-NAME
      FIELD   NAME=(LASTNAME),START=37,BYTES=30,TYPE=C,
              EXTERNALNAME=LAST-NAME
      FIELD   NAME=BIRTHDAY,START=67,BYTES=10,TYPE=C,
              EXTERNALNAME=BIRTHDAY
      FIELD   NAME=PHONE,START=77,BYTES=12,TYPE=C,
              EXTERNALNAME=PHONE
      FIELD   NAME=SEX,START=89,BYTES=6,TYPE=C,
              EXTERNALNAME=SEX
```

Figure 6-2 DBD G2CSTMPR without DBVER

The segment is extended to include a new field for a mobile number. Not all of the existing programs need access to this new field. Some programs might be modified or new programs written. The DBD is modified with the DBVER=1 and a new segment length is defined for CUSTOMER segment. A new copybook is created for the programs that must see the new fields, as shown in Figure 6-3.

```
01 CUSTOMER
   05 CUSTNO        PIC(06)
   05 FIRST-NAME    PIC(30)
   05 CUST-NAME     PIC(30)
   05 BIRTHDAY      PIC(10)
   05 PHONE         PIC(12)
   05 SEX           PIC(6)
   05 MOBILE-PHONE  PIC(12)
```

Figure 6-3 CUSTOMER segment for DBVER = 1

The DBD was modified to use the new DBVER parameter and the version number set to 1, as shown in Figure 6-4.

```
       DBD NAME=G2CSTMRP,ACCESS=(PHIDAM,VSAM),DBVER=1
*
       SEGM    NAME=CUSTOMER,BYTES=106,PTR=TB
       FIELD   NAME=(CUSTSEQF,SEQ,U),START=1,BYTES=6,TYPE=C,
               EXTERNALNAME=CUSTSEQF
       FIELD   NAME=(FRSTNAME),START=7,BYTES=30,TYPE=C,
               EXTERNALNAME=FIRST_NAME
       FIELD   NAME=(CUSTNAME),START=37,BYTES=30,TYPE=C,
               EXTERNALNAME=CUSTOMER_NAME
       FIELD   NAME=BIRTHDAY,START=67,BYTES=10,TYPE=C,
               EXTERNALNAME=BIRTHDAY
       FIELD   NAME=PHONE,START=77,BYTES=12,TYPE=C,
               EXTERNALNAME=PHONE
       FIELD   NAME=SEX,START=89,BYTES=6,TYPE=C,
               EXTERNALNAME=SEX
       FIELD NAME=MOBILE,START=95,BYTES=12,TYPC=C,
               EXTERNALNAME=MOBILE-PHONE
```

Figure 6-4 DBD G2CSTMRP with DBVER=1.

The database is unloaded by using the old DBD and reloaded by using the new DBD to extend the physical length of the segment. The ACBGEN is performed and the DBD is added to the IMS catalog as DBVER=1, which leaves the previous version of 0 still in the IMS catalog.

A new PSB is created for this application by using the DBVER=1 on the PCB statement for the CUSTOMER database. The new program now receives the full CUSTOMER segment of 106 bytes and the programs that are using the PSB without the DBVER specified receive the first 94 bytes of the CUSTOMER segment in the IOAREA as they always have. The PSB that is shown in Figure 6-5 was added to the IMS system and are included in the ACBGEN when the DBDGEN was done for the DBD change.

```
         PCB TYPE=DB,DBDNAME=G2CSTMRP,PROCOPT=AP,
               KEYLEN=100,PCBNAME=XXCSTP,DBVER=1
         SENSEG NAME=CUSTOMER
         SENSEG NAME=DISTRICT,PARENT=CUSTOMER
         SENSEG NAME=CUSTLOCN,PARENT=CUSTOMER
         SENSEG NAME=ADDRLINE,PARENT=CUSTLOCN
         SENSEG NAME=CUSTORDN,PARENT=CUSTLOCN
         SENSEG NAME=CUSTINVN,PARENT=CUSTOMER
         SENSEG NAME=PAYMENTS,PARENT=CUSTOMER
         SENSEG NAME=ADJUSTMT,PARENT=CUSTOMER
*
 PSBGEN LANG=COBOL,PSBNAME=G2ACSTM2,CMPAT=YES
 END
```

Figure 6-5 PSB created for this application by using the DBVER=1

Because the FIELD statement for the MOBILE element was included in the DBD, when programs that are using the DBVER=0 or BASE version of the DBD insert the CUSTOMER segment, the mobile field is filled with blanks. When those same programs retrieve and then update the CUSTOMER segments, the MOBILE FIELD is not passed to the program; therefore, any value in the field is left unchanged after a REPL call.

The programs that were compiled with the new copybook and use the new PSB can enter and update data in the new MOBILE-PHONE field in the database.

6.3 Using database versioning

The IMS version 13 is the only version of IMS that supports databases versioning. The IMS system that is used to support versioning is required to use the IMS catalog. The IMS systems supporting database versioning must have the RECON MINVERS set to “13.1”. An IMS system that is running a lower version of IMS than IMS V13 cannot access versions of a database.

6.3.1 Enabling IMS database versioning

To enable the database versioning support, the required setup is described in this section.

The database that is to be versioned must be physically changed to include the new segment length and new field statements. The new physical structure of the database is called the *current version* and all the previous versions are called by their version number. The original version is called the base version and is referenced by version number of 0.

The database to be versioned uses a new parameter on the DBD statement, DBVER=xxxx, and a normal DBDGEN process is run. A normal unload and reload process can be used to make the physical changes to the segments. All of the IMS utilities always access the current version of the database, including, Image Copy, Recovery, HD Unload, HD Reload, and Prefix Update utilities.

The IMS systems that are going to the support version must specify a new parameter, DBVERSION=Y, in the databases section in the DFSDFxxx member of the IMS PROCLIB data set to indicate that database versioning is to be enabled.

If Database Versioning is enabled, the IMS Catalog must be available to retrieve the correct DBD version for the application programs. If the IMS Catalog is not available and the application is requesting access to any version other that the current version, a “NA” status code is returned to the program.

Application programs that must access a particular version of a database definition can specify the DBVER= on the PCB statement of the PSB source or issue the INIT VERSION call to specify the database version for each database view that is used by the application.

An application program can use the following methods to access a version of the database that is not the current version:

- PSB can set the version to use the current or base version by use of the new parameter on the PSBGEN statement DBLEVEL.
- PCB can use the new parameter DBVER= on individual PCB statements within a PSB.
- A program can issue a new parameter on the INIT DLI call “VERSION DBNAME=nn”.

6.3.2 IMS system changes to enable database versioning

This section describes the changes that were made to enable database versioning.

PROCLIB member changes

In this section, we describe the PROCLIB member changes.

DFSDFxxx DATABASE section

The DFSDFxxx PROCLIB member must have a new parameter in the DATABASE section. The DBVERSION parameter sets the enabling or disabling of versioning for this system. If the DBVERSION=Y is set in the DATABASE section, database versioning is enabled for all the databases that are defined to this system. The only databases that are affected by this setting are those databases for which the catalog has more than one version of the database defined.

The DBLEVEL parameter sets the default for the entire system to use the current or base version for all the databases. Setting the default for the system determines what is done for the application programs. If the intent is to allow the existing Sibs to not be modified at all so that they use the BASE version of the database, the DBLEVEL in the DFSDF member must set to BASE. (This setting is not the default.) If the DBLEVEL is not specified or is specified as CURR, the Sibs that must access the BASE version of a database must be changed to set them to BASE. The example in Figure 6-6 shows the DFSDF database section setup to use BASE as the default.

```
*------------------------------------------------------------------*
* DATABASE  SECTION                                                *
*------------------------------------------------------------------*
<SECTION=DATABASE>
*------------------------------------------------------------------*
DBVERSION=(Y,DBLEVEL=BASE)
```

Figure 6-6 Database section of the DFSDF PROCLIB member

When database versioning is not enabled, IMS does not use anything except the current version of the database as defined by the ACBLIB (even though the IMS catalog can have database with multiple versions).

DFSDFxxx CATALOG section

The number of entries in the IMS catalog and the retention period should be reviewed and modified as required. The catalog keeps a number of instances of the PSB and DBD resources available. These instances are the same version of the PSB or DBD, but with a different ACBGEN time stamp. It specifies the number of instances for each and a retention period (in days) that each instance is maintained.

With Database versioning enabled, the catalog does not keep each DB version as separate instance. The retention number and period now apply to each DB version of the resources. The catalog does not automatically purge resources if the number or retention period is exceeded; however, the Catalog Record Purge Utility must be used to remove expired entries. For more information about the use of this utility, see 5.3.6, “IMS Catalog Record Purge utility” on page 147.

RECON changes

The RECON data set must have the MINVERS parameter set to “13.1” to enable DB versioning, which means that all of the IMS systems in that IMSPLEX must be IMS V13. When a RECON is started by using the V13 SDFSRESL, the default is “11.1”; therefore, you must set it to “13.1”. An example of the change command is shown in Figure 6-7.

```
CHANGE.RECON MINVERS('13.1')
```

Figure 6-7 CHANGE.RECON command to set MINVERS.

6.3.3 DBD changes to enable database versioning

Database versioning is valid only for the following database types:

- DEDB
- HDAM
- HIDAM
- PHDAM
- PHIDAM

A new parameter was added to the DBD statement. The DBVER parameter was added to allow a version number to be set for this database. An example of the DBVER parameter format is show in Figure 6-8.

```
 DBD NAME=G2PRDMRP,ACCESS=(PHDAM,OSAM),RMNAME=(DFSHDC40,4,400),DBVER=1
```

Figure 6-8 DBD statement with the DBVER= parameter

The version number can be any number 1 - 2147483647. It must be a numeric value and it must be a higher value than any version number for this defined database. The number might be a meaningful, such as the Julian date to indicate the date that this change became effective.

After the DBVER parameter is added to the DBD statement, make the required changes to the SEGM statement (or statements) to increase the segment length of the required segments, as shown in Figure 6-9.

```
  DBD NAME=G2CSTMRP,ACCESS=(PHIDAM,VSAM)
 *
  SEGM    NAME=CUSTOMER,BYTES=157,PTR=TB
```

Figure 6-9 DBD and SEGM statement before the DBD change

Figure 6-10 shows the DBD and SEGM statement after the DBVER is added with the new SEGM length.

```
 DBD NAME=G2CSTMRP,ACCESS=(PHIDAM,VSAM),DBVER=1
*
 SEGM    NAME=CUSTOMER,BYTES=211,PTR=TB
```

Figure 6-10 DBD and SEGM statement after the DBVER added with new SEGM length

Defining the fields in the extended part of the segment is not mandatory, but they can be defined as are any other fields. The advantage of defining the fields is that for a program that is not sensitive to those fields, IMS uses some default values to ensure that valid data is placed in those fields. The default value for space that is not defined by fields is binary zeros (x'00').

Note: Variable length segments or segments that are involved in logical relationship are excluded from databases versioning in this version of IMS.

Table 6-1 shows the FIELD types and values used. Setting these fields avoids writing an application program to complete the default value to avoid program abends that are retrieving invalid values in fields.

Table 6-1 FIELD types and values

FIELD TYPE	VALUE	EXCEPTION
C	X'40'	N/A
P	X'00'	LOWER ORDER BYTE X'0C'
X	X'00'	N/A

After the source of the DBD is changed, the normal process to implement a DBD change where the structure changed is required. You must unload the database with the previous version of the DBD and reload with the modified version of the DBD. If the DBD that is modified includes secondary indexes or logical relationships, the normal utilities that are used in the reorganization process are still required. If the DBDs are HALDBs, the secondary indexes do not need to be involved in the reorganization process.

If the DBDs that are modified include logical relationships, all of the related databases must be included in the ACBGEN with BUILD statements for them all.

6.3.4 PSB changes to enable database versioning

The PSB can be used to determine which version of a database this PSB uses. This determination can be done at the PSB level or individual PCB levels.

The PSB can set the default level for all the PCBs within this PSB to CURR or BASE. This setting overrides the DBLEVEL parameter that was set in the DFSDF member for this system. An example of the two options is shown in Figure 6-11.

```
PSBGEN PSBNAME=psbname,DBLEVEL=CURR

PSBGEN PSBNAME=psbname,DBLEVEL=BASE
```

Figure 6-11 PSBGEN with DBLEVEL specified

If the DBLEVEL is not specified, the DBVERS parameter is used on individual PCBs. If the DBLEVEL and a DBVERS are coded on the PSBGEN and PCB statements, the DBVERS overrides the PSBGEN setting.

Individual PCB statements can be coded with the DBVERS parameter to specify the exact version of a database to be used for this PCB (see Figure 6-12). If there are multiple PCBs for the same DBD within this PSB, all of the PCB statements must name the same DBVERS value, which means that a program can use only one database version. It cannot read or write multiple versions of the same database. The PSB uses the DBVERS parameter to define the version of the database for this PCB. The program can optionally use the AIB interface with a PSBNAME or LABEL, but it is not a requirement for database versioning.

```
label PCB TYPE=DB,DBDNAME=dbdname,DBVERS=1,PCBNAME=pcbname
```

Figure 6-12 PCB statement with DBVERS parameter

For those PSBs that are not modified to use the DBVER or DBLEVEL the IMS system default is used. If the system default is set to base, all of the older PSBs continue to use the version of the database they always used. If the IMS system default is set to CURR, all programs that do not explicitly set the database version to base in the program or PSB now receives the extended segment layouts. If those programs were not recompiled with the new copybooks to get the increase segment length, unpredictable results might occur.

6.3.5 ACB changes to enable database versioning

The are no specific changes that are required to the ACBGEN process to enable database versioning. The ACBs must be populated into the IMS catalog by using the new ACBGEN process that is implemented in version 12 of IMS or the catalog populate utility after the ACBGEN.

There is a new utility that is called the *Catalog Populate Utility*. It can be used to load the catalog or insert new entries into the catalog. It runs as a batch process and reads the ACBLIB as input. All of the members in the ACBLIB are copied to the catalog. The IMS catalog can maintain multiple entries for ACBLIB members. Each entry is based on the member name and a time stamp that is created at ACBGEN time. If the Catalog Populate utility is run in “insert” mode with a PSB with a PROCOPT other than L, entries with the existing time stamp that corresponds to the time stamp in the ACBLIB member are updated. If the ACBLIB members have different time stamps from the catalog entries, they are added.

The use of the DFSCPL00 PSB to update the IMS catalog must be done in batch mode. The IMS catalog must be available to be authorized by at DLIBATCH job. The DLIBATCH job must have DBRC=Y and a valid IEFRDER log file specified. Therefore, if you are not data sharing the IMS catalog via the IRLM to DLIBATCH jobs, the Catalog must be offline from the online systems. After the utility runs in this mode, the IC NEEDED flags must be updated in the RECON for the catalog. The log file also is recorded in the RECON for recovery situations. The DLIBATCH job must also name the DFSDFxxx member in the IMS.PROCLIB library in the EXEC parameter, as shown in Figure 6-13 on page 191.

```
//LOADCAT EXEC PGM=DFS3PU00,REGION=7M,
// PARM=(DLI,DFS3PU00,DFSCP000,,,,,,,,,,,Y,N,,,,,,,,,,,,'DFSDF=IMG')
//STEPLIB  DD  DSN=IMS.IMSG.SDFSRESL,DISP=SHR
//         DD  DSN=IMS.IMOD.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.IMOD.SDFSRESL,DISP=SHR
//IMS      DD  DSN=IMS.IMSG.PSBLIB,DISP=SHR
//         DD  DSN=IMS.IMSG.DBDLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.IMSG.PROCLIB,DISP=SHR
//SYSABEND DD  SYSOUT=*       DUMP DATA SET
//SYSPRINT DD  SYSOUT=*       MESSAGES, STATISTICS
//IEFRDER  DD  DSN=IMS.IMSG.SLDS(+1),DISP=(,CATLG),
//    UNIT=SYSDA,SPACE=(CYL,(10,10),RLSE)
//DFSVSAMP DD  DSN=IMS.IMSG.UTIL(VSAMP),DISP=SHR
//IMSACB01 DD  DSN=IMS.IMSG.ACBLIB,DISP=SHR
```

Figure 6-13 Inserting new entries into the catalog by using a PROCOPT=A

To load the IMS catalog when it is empty or to refresh all of the entries in the catalog, run the catalog utility with the DFSCPL00 PSB. The example in Figure 6-13 on page 191 uses the DFSCP000 PSB, which inserts new entries into the catalog by using a PROCOPT=A. The utility can support multiple ACBLIB data sets by adding DD statements with a numeric value after the ACBLIB in the DD statement.

The alternative to the use of the catalog populate utility is to use the modified ACBGEN process. A new ACBGEN procedure that is called DFS3UACB is a combined ACBGEN and catalog populate utility. This procedure runs the ACBGEN program, then updates the IMS catalog via a BMP step. Only the ACBLIB members that were generated in this run are populated to the IMS catalog.

The DFS3UACB procedure is shown in Figure 6-14. The procedure must be modified to use the correct libraries, data sets names, and BMP parameters to attach to the online system.

```
//ACBCATT EXEC PGM=DFS3UACB,REGION=6M
//STEPLIB  DD  DSN=IMS.IMSG.EXITLIB,DISP=SHR
//         DD  DSN=IMS.IMSG.SDFSRESL,DISP=SHR
//DFSRESLB DD  DSN=IMS.IMSG.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.IMSG.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//IMS      DD  DSN=IMS.IMSG.PSBLIB,DISP=SHR
//         DD  DSN=IMS.IMSG.DBDLIB,DISP=SHR
//*******************************************************************
//* ACBGEN DATASETS
//*******************************************************************
//IMSACB   DD  DSN=IMS.IMSG.ACBLIB,DISP=SHR
//SYSUT3   DD  UNIT=SYSDA,SPACE=(80,(100,100))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(256,(100,100)),DCB=KEYLEN=30
//SYSIN    DD  *
  BUILD  DBD=G2CSTMRP
  BUILD  DBD=G2PRDMRP
  BUILD  PSB=G2ACSTLD
  BUILD  PSB=G2PUTCS1
//ACBCATWK DD  DSN=USER.IMDD.ACBCATWK,DISP=(NEW,PASS),
//             SPACE=(CYL,(10,10)),unit=3390,
//*            DCB=(BLKSIZE=800,RECFM=FB,LRECL=80)
//******************************************************************
//* POPULATE UTILITY DATASETS
//*******************************************************************
//IMSACB01 DD  DSN=*.IMSACB,DISP=SHR      DO NOT REPLACE ASTERISK
//DFSVSAMD DD  DSN=IMS.IMDD.UTIL(VSAMP),DISP=SHR
//*******************************************************************
//* UPDATE INPUT PARMS FOR IMS CATALOG POPULATE UTILITY
//*******************************************************************
//DFS3PPRM DD  *
BMP,DFS3PU00,DFSCP001,,,N00000,,,,1,,,,IMSG,BMP01,,,,'',
/*
//SYSINP DD *
ISRTLIST
DUPLIST
/*
//
```

Figure 6-14 Example DFS3UACB Procedure

The IMS catalog also must be opened for update processing in the IMS system. The catalog is not started in update mode by default; therefore, explicit commands must be used to start it in update mode, as shown in Figure 6-15.

```
//SPOC      EXEC PGM=CSLUSPOC,
//  PARM=('IMSPLEX=PLX13,ROUTE=IMSG,WAIT=30')
//STEPLIB   DD DISP=SHR,DSN=IMS.IMSG.SDFSRESL
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 UPDATE DB NAME=(DFSC*) START(ACCESS) SET(ACCTYPE(UPD))
 QRY DB NAME=(DFS*) SHOW(STATUS)
//
```

Figure 6-15 Start commands to put IMS catalog in UPDATE mode

The staging ACBLIB contains only the ACTIVE or most recent version of the DBD or PSB. The IMS catalog is the only place that previous versions of the resources are kept. It becomes critical that the catalog is backed up and procedures are in place to recover it if needed. The catalog is only a HALDB database; therefore, all updates are logged as with any other database updates. These logs must be retained until the next Image Copy Utility run or they can be accumulated in a CA group. If you use the DFS3UACB procedure on multiple IMS systems, you must use the CA process to merge those updates before a recovery can be run.

6.3.6 Program changes to enable database versioning

Individual programs can use a new feature of the INIT call to set the database version for one or all of the databases to which this program is sensitive. The use of the INIT call to set database version overrides the system default and the DBVERS or DBLEVEL PSB specifications. The INIT call can specify a version for one or more than one DBD, as shown in Figure 6-16. The size of the IOAREA that is required in the VERSION call is calculated by multiplying 20 bytes by the number of DBDNAMES supplied.

```
77 INIT   PIC X(4)  VALUE ‘INIT’.
77 IOAREA PIC X(40).

MOVE ‘VERSION(dbdname=xx,dbdnam2=yy)’ TO IOAREA

CALL 'CBLTDLI' USING INIT, IOPCB, IOAREA.
```

Figure 6-16 INIT DLI call with VERSION keyword specified

The value in the IOAREA must meet the following strict requirements:

- The VERSION keyword must be in the first 7 bytes
- An open parenthesis must follow the VERSION keyword
- DBDNAME must be a valid DBDNAME
- An equal sign must follow the DBDNAME without any spaces
- The version number must be numeric
- A comma is used to supply multiple DBDNAMES
- A close parenthesis must be at the end
- Duplicate DBDNAMES cannot be used
- The DBDNAME must name the physical DBD name not a LOGICAL DBD

As with all INIT calls, this INIT call must precede any DLI call in the program. Multiple INIT calls are allowed in a program. You can include a DBD in multiple calls; however, if you do include multiple calls, the version must always be the same number. There is no reason to use multiple INIT calls for the same DBDs. You can also reduce the number of INIT calls by using multiple Dependants in the call.

If the program issues a GU to the IOPCB, this GU call should proceed the INIT calls for performance reasons.

There are several new status codes that can be returned to a program that is issuing an INIT call with the VERSION parameter, as listed in Table 6-2.

Table 6-2 Status codes that are returned to INIT call with the VERSION parameter

Status Code	Reason	Action
BE	Version number that is specified on the INIT call cannot be found in the IMS catalog.	Correct the version number on the INIT call.
BF	Reasons for this status code: ▶ Invalid version number for this database ▶ Version number not in the support range ▶ Database versioning not enabled in this IMS system ▶ Different version number than a previous INIT call is specified for this database	Change the version number to the correct number for this DBD. Change the IMS DFSDFxxx member to support database versioning. Ensure the save version number is used for a DBD on multiple INIT calls.
BG	GSAM, MSDB, or logical DBD name that is used in the INIT call	Correct the DBD to a support DBD type

IMS system that is running with DB versioning active

There are no changes to the IMS online system except to enable versioning in the DATABASE section of the DFSDFxxx member and installing the IMS catalog. After the DATABASE is set and catalogs are installed, the database versioning is now active.

The system operates as normal until the first ACBGEN brings in a DBD and PSB with a DBVER. IMS reads the ACBLIB for all the CURRENT DBDs and DBDs and builds control blocks with the relevant information. For the first transaction or program that requests a PSB or DBD that is not the CURRENT, the control blocks are built by using information from the IMS catalog instead of the ACBLIB. Those control blocks are used for all subsequent access. The control blocks are maintained until the next cold start of this system. This process is shown in Figure 6-17 on page 195.

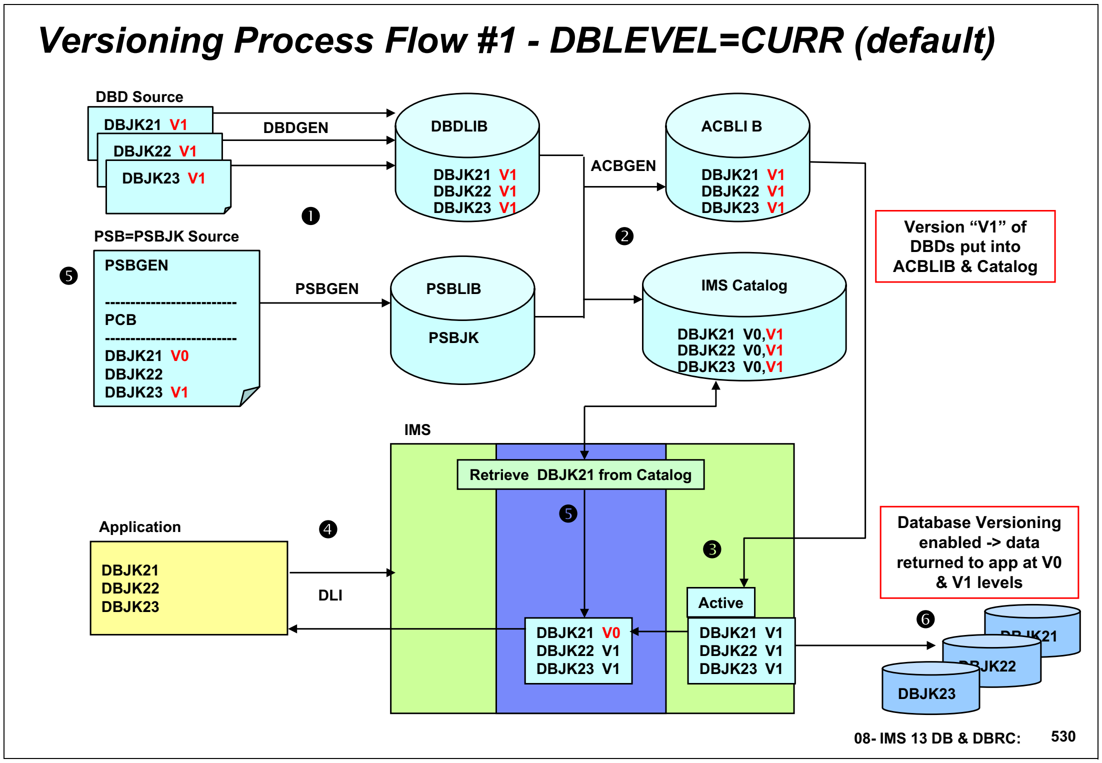


Figure 6-17 IMS catalog that is accessed in an online system

Those programs that are retrieving segments from a database other than the current version have the segments modified in the buffer pool to return or update only the relevant part of the segment for this version.

Because the IMS catalog is a HALDB database, all the access processes through the normal buffer pools specifications for a database. The system should have enough buffers of the appropriate type to ensure that the access is as efficient as possible.

The catalog is not updated, except by the ACBGEN process. Unless the DFS3UACB process is used, the catalog database can be opened in READ mode. Each IMS system in an IMSPLEX accesses the catalog as needed for programs that are running in that IMS system.

7

Application programming enhancements

IMS Version 13 provides several enhancements for application programs. IMS application programs can use the following enhancements that are described in this chapter:

- IMS synchronous callout support
- Using SQL in COBOL to access your IMS databases

The following functions also help application developers in IMS 13:

- Database versioning (for more information, see Chapter 6, “Database versioning” on page 181)
- ESAF support in Java dependent regions (for more information, see 2.9, “ESAF support in Java dependent regions” on page 62)

This chapter includes the following topics:

- IMS synchronous callout support
- Using SQL in COBOL to access your IMS databases

7.1 IMS synchronous callout support

In this section, we introduce the synchronous callout support in IMS and show an example of its use in a COBOL program.

We then describe enhancements to the related ICAL DL/I call as provided by IMS 13.

7.1.1 Callout applications

As Internet technology evolves and business requirements gain complexity with speed, businesses are calling for better exchanges and integration of data and services. It is now a common scenario in which IMS applications are required to access data or services from sources outside of IMS, z/OS, or the company intranet to complete a task.

With the IMS callout solutions, IMS applications can send outbound messages (through Open Transaction Manager Access [OTMA] and IMS Connect) to request data or services from an external application, such as a Web service, a message-driven bean (MDB), or an Enterprise JavaBeans (EJB) application. The request for external data or services is called an IMS callout request. The application that issues a callout request is an IMS callout application.

An IMS callout application can receive responses back in the same or a different application. An IMS callout application can handle the response in two ways: asynchronously or synchronously. If the callout application frees up the dependent region and does not wait for a response after the request is issued, it is called an asynchronous callout request. If it waits for that response, it is a synchronous callout request.

In many business cases, synchronous callout requests are required because the callout application must receive some data back (or an acknowledgment of successful delivery) to continue with subsequent operations.

7.1.2 Application integration with synchronous callout

Many complex business problems require that IMS applications be integrated with external services or applications. When an IMS callout application must wait for the response before it can complete a task or proceed with subsequent operations, the IMS synchronous callout function provides the technical solution.

To issue a synchronous callout request in an IMS application program, use the ICAL DL/I call. Each call must specify an AIB to communicate with IMS with detailed information, such as the request destination (which is the name of an OTMA destination descriptor) and a timeout value. The entire AIB structure must be defined in working storage, on a fullword boundary, and initialized according to the order and byte length of the fields.

For more information about implementing the synchronous callout function, see *IMS Version 13 Application Programming*, SC19-3646.

An example of the use of COBOL is shown in 7.1.3, “Using IMS Application Interface Block in COBOL for synchronous callout” on page 199.

Samples of programs with synchronous callout in REXX and PL/I are described in the article “How to Code a Synchronous Program Switch in IMS 13”, which is available at this website:

http://www.ibmsystemsmag.com/mainframe/administrator/ims/ims_program_switch/

7.1.3 Using IMS Application Interface Block in COBOL for synchronous callout

Each ICAL DL/I call in an IMS application program must specify an application interface block (AIB) to communicate with IMS with the detailed instructions for calling out. Some of the information in the AIB is passed to an IMS Connect client, such as IMS Enterprise Suite SOAP Gateway or WebSphere DataPower® Appliance, for callout message formatting. Each AIB in a COBOL application consists of several AIB fields and must be defined in working storage, on a fullword boundary, and initialized according to the order and byte length of the AIB field. The use of the AIB correctly is the first step to making an IMS callout request for a COBOL application.

Two of the common fields in the AIB are the specifications of the request destination and a timeout value. The ICAL call must be provided with these AIB fields to determine where to send the request and, if the default timeout value is not appropriate, how long it must wait for the response.

The information that is specified in the AIB is the first level of instructions for calling out. The second level is defined in the OTMA destination descriptor in the IMS DFSYDTx procedure library member. The AIB must specify the name of an OTMA destination descriptor so IMS can locate the specified OTMA descriptor entry to obtain the address of the callout request, as shown in Figure 7-1.

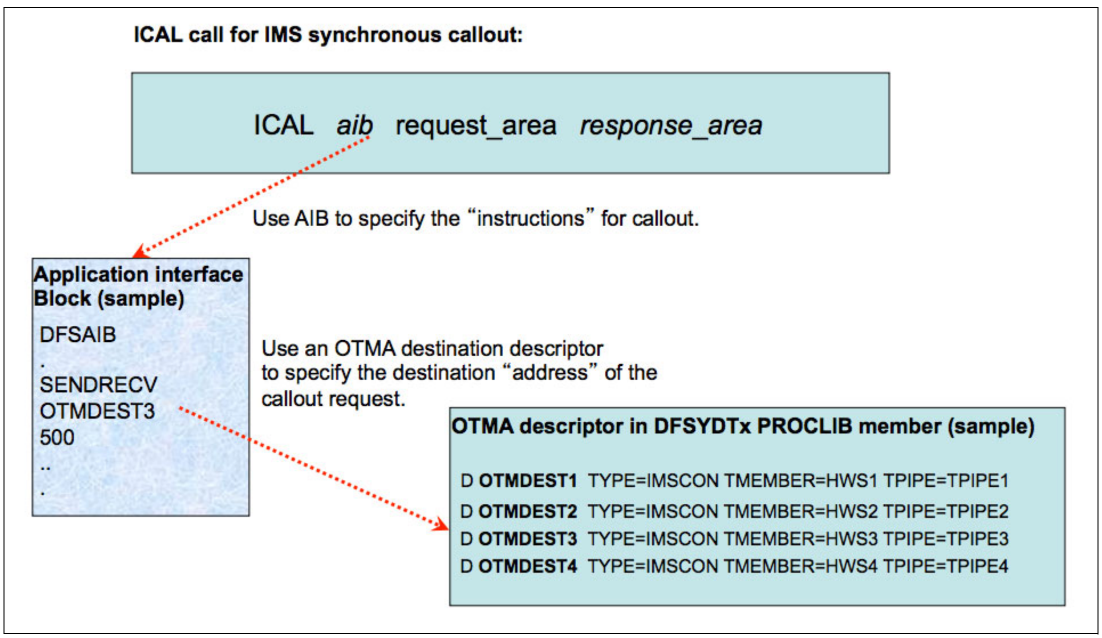


Figure 7-1 Relationship between AIB and OTMA descriptor for IMS callout

If the AIB does not have the valid information that is specified, the callout request is rejected immediately. The return code, reason code, and extended reason code fields inside the AIB explain the result of the callout request.

The complete list of AIB fields for the ICAL call is shown in Table 7-1.

Table 7-1 AIB Fields for the ICAL Call

Fields	Byte Length	Description
AIBID	3	Eyecatcher, DFSAIB
AIBLEN	4	Length of AIB. This is a required input field.
AIBSFUNC	8	Subfunction code, SENDRECV or RECEIVE. The SENDRECV is used for most cases for calling out or synchronous program switch. The RECEIVE code is introduced in IMS V13 to receive the response data. This is a required input field.
AIBRSNM1	8	Destination name of OTMA destination descriptor, as shown in Figure 7-1 on page 199. This is a required input field.
AIBRSNM2	8	LTERM override for synchronous program switch, which is not to be used for synchronous callout. This is an optional input field.
RESERVE D	8	Reserved field.
AIBOALEN	4	Callout request area length. This is an input and output field for the SENDRECV subfunction. As an input parameter, this 4-byte field must contain the length of the input request area that is specified in the call list. As an output parameter, this field is updated only when partial data is returned (AIB return code X'100', AIB reason code X'00C'). When partial data is returned, this field contains the actual length of the response message. For any other return codes, this field is unchanged. When this filed is used for the RECEIVE subfunction, it is used as an output parameter. When the complete response data is returned in the response data area, this output field is set to 0. When partial data is returned (AIB return code X'100', AIB reason code X'00C'), this field contains the actual length of the response message.
AIBOAUSE	4	Callout response area length. This is an input and output field for the SENDRECV and RECEIVE subfunction. As an input parameter, this 4-byte field must contain the length of the output response area that is specified in the call list. As an output parameter, this field is updated by IMS with the length of the response message that is returned in the response area. When partial data is returned because the response area is not large enough, this field contains the length of the data that is returned in the response area, and AIBOALEN contains the actual length of the response message.
AIBRSFLD	4	Timeout value in 100th of a second. This is an optional input field. If the timeout value is specified on the OTMA destination descriptor and on the ICAL call, IMS uses the lesser value of the two. Ten seconds is used as the default if nothing is specified.
RESERVED	8	Reserved field.
AIBRETRN	4	AIB return code. This is an output field.
AIBREASN	4	AIB reason code. This is an output field.
AIBERRXT	4	AIB extended reason code. This is an output field.
RESERVED	12	Reserved field.
AIBUTKN	8	User-specified map name for callout. This is an optional input field for message formatting or service identification purposes.

Example 7-1 shows how to use the AIB interface in a COBOL application for synchronous callout.

Example 7-1 AIB interface in a COBOL application

```
*
      * PROCEDURE ICAL-CALLOUT
      * Setup to make an ICAL call
      * - Set AIBLEN, AIB Subfunction (SENDRECV), OTMA destination descriptor
      * - Set wait time to AIBRSFLD
      * - Set length of request and response areas to AIBOALEN and AIBOAUSE
       ICAL-CALLOUT.
           MOVE LENGTH OF AIB TO AIBRLEN.
           MOVE "SENDRECV" TO AIBSFUNC.
           MOVE "OTMDEST3" TO AIBRSNM1.
           MOVE 500  TO AIBRSFLD.
           MOVE LENGTH OF CALL-REQUEST TO AIBOALEN.
           MOVE LENGTH OF CALL-RESP TO AIBOAUSE.
           MOVE AIBOALEN TO  AIBOALEN-EDITED.
           MOVE AIBOAUSE TO  AIBOAUSE-EDITED.
           DISPLAY 'REQUEST  AREA    : ' AIBOALEN-EDITED UPON CONSOLE.
           DISPLAY 'RESPONSE AREA    : ' AIBOAUSE-EDITED UPON CONSOLE.
      *
      *  CALL ICAL
      *
           CALL 'AIBTDLI' USING ICAL, AIB, CALL-REQUEST, CALL-RESP.
```

An ICAL-callout procedure is created by setting up the AIB length, subfunction, OTMA destination descriptor name, timeout value, and the lengths of the callout request and response. Then, the ICAL DL/I call is issued by using the following AIBTDLI interface format:

```
CALL 'AIBTDLI' USING function, aib, i/o area
```

7.1.4 ICAL enhancements

IMS 13 introduces enhancements to ICAL synchronous callout support by providing a new RECEIVE sub function that enhances the support for truncated messages. Also, a new AIB field, AIBUTKN, is added. By using AIBUTKN, users can send a name to a remote ICAL destination that can be used for message formatting or service identification purposes.

Support for truncated messages

IMS 13 introduces a new sub function of RECEIVE for the ICAL. This enhancement supports the retrieval of an entire response message if the original ICAL received partial data and an AIB return code of x'100' and reason code of x'00C' because of inadequate application specification. By using the new sub function and expanding the response area in the call, the IMS application can retrieve the copy of the message that is temporarily kept in the IMS control region's private storage. The message is kept in the memory until a subsequent ICAL SENDRECV, syncpoint, or application ending.

This enhancement relieves the burden of having to reissue a full ICAL SENDRECV along with any associated network delays.

Figure 7-2 shows the format of ICAL sub function RECEIVE call.

ICAL subfunction RECEIVE

- Format:
  - >>-ICAL--*aib--response area*---------------------------------------><
- AIB
  - *AIBSFUNC* value "RECEIVE"
  - *AIBOAUSE* is used as an input and output parameter based on AIBSFUNC
    - For the "RECEIVE" call
      - Contains the length of the response area
  - *AIBOALEN* = request area length
    - Used as an *output parameter for "RECEIVE"*
      - When complete response is returned in *response area*, this field is 0
      - If partial data is returned (AIB RC X'100', RS X'00C'), this field contains the actual length of the response message

Figure 7-2 ICAL sub function RECEIVE

The format of the ICAL with a sub function of RECEIVE includes specification of the AIB and a response area that can contain the entire message. Not all of the AIB fields are used for this call. The following parameters apply:

- `AIBID = DFSAIB`
- `AIBLEN = AIB length`

AIBOALEN is the request area length. This parameter is an input and output parameter for the SENDRECV subfunction. When this parameter is used for the RECEIVE subfunction, it is used as an output parameter. When the complete response data is returned in the response data area, this output field is set to 0. When partial data is returned (AIB return code X'100', AIB reason code X'00C'), this field contains the actual length of the response message.

AIBOAUSE is the area length. This parameter is an input and output parameter. As an input parameter, this 4-byte field contains the length of the output response area that is specified in the call list. As an output parameter, this field is updated by IMS with the length of the response message that is returned in the response area. When partial data is returned because the response area is not large enough, AIBOAUSE contains the length of the data that is returned in the response area, and AIBOALEN contains the actual length of the response message.

The following AIB fields contain error message information:

- `AIBRETRN = AIB Return code`
- `AIBREASN = AIB Reason code`

If an ICAL call with subfunction code SENDRECV is issued with an insufficient response buffer defined, the ICAL is returned with AIB return code X'100' and AIB reason code X'00C'. A subsequent ICAL call with subfunction code RECEIVE can still be issued to retrieve the entire response message if it is defined with the expanded response buffer, as shown in Figure 7-3 on page 203.

ICAL sub-function RECEIVE …

- Usage example:
  - **ICAL --*aib—request area, response area***
    - AIBSFUNC (SENDRECV)
    - AIBOAUSE – Response area length
    - CALL is issued → AIBRETRN=x'100', AIBREASN='00C'
      - Specified length of the output response area is too small
      - AIBOAUSE= length of the data that was returned in the response area
      - AIBOALEN = the actual length of the entire response message
    - Using the value in the previous AIBOALEN and leveraging the new support which keeps the message in IMS CTL region private, retrieve the entire response:
  - **ICAL --*aib— response area***
    - Where *response area* has been expanded to contain the entire message
    - AIBSFUNC (RECEIVE)
    - AIBOAUSE – new response area length
    - CALL is issued successfully
      - ✓ AIBOAUSE – length of the response in the response area
      - ✓ AIBOALEN – set to 0 because the call successfully returned the entire response

Figure 7-3 ICAL sub function RECEIVE usage example

To issue an ICAL call with the subfunction code RECEIVE to retrieve the entire response message, the ICAL call with subfunction code SENDRECV must be issued and returned with AIB return code X'100' and AIB reason code X'00C'.

IMS keeps a copy of the entire response message in the control region private for any potential subsequent ICAL RECEIVE calls. If this copy of the response message in IMS is not retrieved by the ICAL RECEIVE call, it is discarded and the storage is released. The response message is kept until the following events occur:

- A new ICAL call with subfunction code SENDRECV is issued.
- The IMS application issues ROLB call or CHECKPOINT.
- The IMS application reaches syncpoint or ends abnormally.

New AIBUTKN parameter

The DL/I ICAL call for synchronous callout can optionally use the first 8 bytes of a new AIBUTKN parameter to specify a map name in the application interface block (AIB). This value is passed in the OTMA state data prefix and can be used by the remote destination for message formatting or service identification purposes.

This enhancement is also retrofitted to IMS 12 with APAR PM73135.

7.2 Using SQL in COBOL to access your IMS databases

Native SQL support for COBOL in IMS 13 enables COBOL application programs to access IMS databases by using the Structured Query Language (SQL). This support complements the SQL that is available to Java based IMS applications and expands IMS database usage to a wider group of application and database developers who can use SQL skills without requiring in-depth IMS database knowledge.

The Native SQL support is part of the base for IMS V13; therefore, it is available to all IMS users.

7.2.1 Introduction to Native SQL access to IMS databases

The structure of a COBOL application that is accessing databases with the SQL was designed to be as close as possible when the database is IMS as it is when the database that is accessed is DB2. This configuration simplifies the process of accessing data in an IMS database for an application programmer who has experience with SQL accessing DB2.

The COBOL application requires the same artifacts in place when SQL is used as it does when DL/I is used to access the database. The program continues to use a Program Specification Block (PSB) to communicate with IMS, and the system service calls (such as sync) continue to use the DL/I interface. Therefore, most programs must include DL/I calls in addition to SQL access to the databases.

7.2.2 Preparation: Enabling your IMS system for Native SQL

There are several functions that must be enabled in your IMS system to use Native SQL with IMS. The metadata for the databases to be accessed with SQL must be available in the Catalog. Therefore, the IMS system must be running with several of the Common Service Layer address spaces (the Structured Call Interface and Operations Manager as a minimum). The Open Database Manager address space is also needed, which requires IMS Connect.

For more information about the process of capturing the extra database fields into the IMS database description (DBD), see 11.3, “Using IMS Explorer to capture IMS metadata” on page 328.

7.2.3 Software prerequisites

The base IMS V13 code includes the ability to run programs that are accessing IMS databases through SQL. That is, no other products are required.

However, the compilation of applications that are accessing databases with COBOL requires Enterprise COBOL Version 5.1, with the IMS coprocessor maintenance installed. This prerequisite might require an upgrade to the version of compiler that is used.

7.2.4 Native SQL environment

Programs that are accessing IMS databases with Native SQL can be run as Message Processing Programs (MPPs), Interactive Fast Path (IFP) programs, and Batch Message Processing (BMP) programs. The IMS control region must be a full IMS TM/DB system. As of this writing, there is no support for IMS Batch or IMS Database Control (DBCTL) environments.

7.2.5 Running in a mixed DL/I and SQL environment

Applications can continue to access all IMS databases without any modifications.

New applications can be written by using Native SQL to access IMS databases. These new applications can coexist with the current DL/I-based applications without any extra considerations (above those considerations that must be addressed when any other applications are accessing your IMS databases).

An application program can use the DL/I interface and the SQL interface in the same program, and even to the same set of databases.

7.2.6 Compilation of Native SQL programs

The SQL extensions to the COBOL language were implemented in Enterprise COBOL Version 5.1. Any program that is accessing an IMS database with SQL must be compiled with at least this level of the COBOL compiler.

Note: The IMSSQL compiler option is necessary for Native SQL applications.

7.2.7 Running Native SQL programs

When an application starts using the SQL interfaces, the Catalog database is implicitly accessed to provide the database metadata. This task is performed by IMS through the DFSCAT00 Program Communication Block (PCB), which was dynamically added to the PCBs in the PSB that is used by the program since IMS 12.

The standard IMS control blocks that are needed for application access to IMS databases continue to be necessary, whether the application is using Native SQL. That is, DBDs, PSBs, and ACBs are necessary for COBOL programs that are accessing IMS databases, irrespective of which of the DL/I or SQL (or both) interfaces is being used by the application program.

Native SQL programs can be run only as Message Processing Programs (MPPs), Interactive Fast Path Programs (IFPs), and Batch Message Programs (BMPs). The Job Control Language (JCL) that is used to run the application is no different.

7.2.8 IMS SQL statements for COBOL

The following SQL statements can be used to access IMS databases and are supported by IMS:

- CLOSE

  The CLOSE statement closes a cursor.
- DECLARE CURSOR

  The DECLARE CURSOR statement defines a cursor.
- DECLARE STATEMENT
- The DECLARE STATEMENT statement is used for application program documentation. It declares names that are used to identify prepared SQL statements.
- DELETE

  The DELETE statement deletes rows from a table.

- DESCRIBE OUTPUT

  The DESCRIBE OUTPUT statement obtains information about a prepared statement.
- EXECUTE

  The EXECUTE statement runs a prepared SQL statement.
- FETCH

  The FETCH statement positions a cursor on a row of its result table. It can return zero or one and assigns the values of the rows to host variables if there is a target specification.
- INCLUDE

  The INCLUDE statement inserts application code (including declarations and statements) into a source program.
- INSERT

  The INSERT statement inserts rows into a table.
- OPEN

  The OPEN statement opens a cursor so that it can be used to process rows from its result table.
- PREPARE

  The PREPARE statement creates an executable SQL statement from a string form of the statement. The character-string form is called a *statement string*. The executable form is called a *prepared statement*.
- SELECT

  The SELECT statement is used to retrieve data from one or more tables. The result is returned in a tabular result set.
- UPDATE

  The UPDATE statement updates the values of specified columns in rows of a table.
- WHENEVER

  The WHENEVER statement specifies the host language statement to be run when a specified exception condition occurs.

7.2.9 Annotations on a sample IMS SQL application

The section includes a sample COBOL application that uses Native SQL to access data from an IMS database. The program source is shown Example 7-2 on page 206. There are annotations to describe the new SQL and existing IMS function in this application.

All Native SQL applications require some DL/I calls, even if it is used to take only checkpoints to commit database updates.

Example 7-2 Sections of a Native SQL COBOL program

```
 CBL SQLIMS
                             * See “Enterprise COBOL compiler options” on page 210
 IDENTIFICATION DIVISION.
 PROGRAM-ID. 'sqla11'.
- - - - - - - - - - - - - - - - -      Line(s) not Displayed
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  IBM-370.
 OBJECT-COMPUTER.  IBM-370.
```

```
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.

- - - - - - - - - - - - - - - - -      Line(s) not Displayed
*****************************************************
* SQL INCLUDE FOR SQLCA                             *
*****************************************************
                                   * See "IMS SQL Communications Area" on page 210
     EXEC SQLIMS
      INCLUDE SQLIMSCA
     END-EXEC.

*****************************************************
* SQL INCLUDE FOR SQLDA                             *
*****************************************************
                                       * See "IMS SQL Descriptor Area" on page 210
     EXEC SQLIMS
      INCLUDE SQLIMSDA
     END-EXEC.

*****************************************************
* customer information *
*****************************************************
                              * See "Working Storage for IMS SQL data" on page 210
   01 SQL-custroot.
     03 sql-customernumber     PIC S9(11) binary.
     03 sql-firstname          pic x(10).
     03 sql-lastname           pic x(20).
     03 sql-customerdata       pic x(67).

   01  SQL-STATEMENT.
      03 SELECT-STATEMENT-LEN  PIC S9(4) COMP VALUE +200.
      03 STATEMENT-TXT         PIC X(200).
- - - - - - - - - - - - - - - - -      Line(s) not Displayed
 LINKAGE SECTION.
              * See "Linkage section and procedure division statement" on page 211

 01  IOPCB.
- - - - - - - - - - - - - - - - - -   Line(s) not Displayed

 01  IOPCB2.
- - - - - - - - - - - - - - - - - -   Line(s) not Displayed

 01  DBPCB.
- - - - - - - - - - - - - - - - - -   Line(s) not Displayed
PROCEDURE DIVISION USING IOPCB, IOPCB2, DBPCB.
- - - - - - - - - - - - - - - - - -   Line(s) not Displayed
 MAIN-RTN.

     DISPLAY "Cobol Program sqla11 execution begins... ".

- - - - - - - - - - - - - - - - - -   Line(s) not Displayed
                                            * See "Whenever statement" on page 211
     EXEC SQLIMS
```

```
         WHENEVER SQLERROR GOTO 100-DBERROR
       END-EXEC.
       EXEC SQLIMS
         WHENEVER SQLWARNING GOTO 200-DBERROR
       END-EXEC.
       EXEC SQLIMS
         WHENEVER NOT FOUND CONTINUE
       END-EXEC.

-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -     Line(s) not Displayed
                                             * See “Declare a cursor” on page 211
       EXEC SQLIMS
         DECLARE TELE1 CURSOR FOR DYSQL
       END-EXEC.

-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  Line(s) not Displayed
                                           * See “SQL statement text” on page 211
          MOVE "SELECT customernumber FROM PCB01.custroot"
            TO STATEMENT-TXT.

          move 198 to SELECT-STATEMENT-LEN.

-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  Line(s) not Displayed
                                    * See “IMS SQL prepare statement” on page 211
          EXEC SQLIMS
            PREPARE DYSQL FROM :SQL-STATEMENT
          END-EXEC.

                            * See “IMS SQL describe output statement” on page 211
          move 10 to SQLIMSn.
          exec SQLIMS
            describe output dysql into :SQLIMSda
          end-exec.

                       * See “Displaying the SQL IMS descriptor area” on page 213
          perform display-SQLda.

-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  Line(s) not Displayed

                                              * See “Open the cursor” on page 212
          EXEC SQLIMS
            OPEN TELE1
          END-EXEC.

          COMPUTE DBD-COUNT = 0.

*         display "about to perform fetch-CUSTROOT"
          PERFORM FETCH-custroot
             10 times.
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  Line(s) not Displayed

                                             * See “Close the cursor” on page 212
          display "about to perform CLOSE TELE1".

          EXEC SQLIMS
```

```
           CLOSE TELE1
         END-EXEC.

      PERFORM RETURN-TO-CALLER.

  FETCH-CUSTROOT.
- - - - - - - - - - - - - - - - -   Line(s) not Displayed
                                              * See “Fetch the results” on page 213

      EXEC SQLIMS
        FETCH TELE1 INTO :sql-customernumber
      END-EXEC.

      IF SQLIMSCODE EQUAL 0
         ADD 1 TO DBD-COUNT
         display " count ", dbd-count,
                 ", custno= ", sql-customernumber
      end-if.
************************************************************
*    Display Data Not Found Message when data not found.
************************************************************
                                                * See “Data not found” on page 213
  DATA-NOT-FOUND.

      DISPLAY " ----------------------------------------".
      DISPLAY " ".
      DISPLAY "******NO MORE DATA FOUND******".
*     MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
*     DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.

*
************************************************************
*    Display error message
************************************************************
  100-DBERROR.
                                                 * See “SQL IMS error” on page 213
      DISPLAY "ERROR OCCURRED IN WHENEVER SQLIMSERROR".
      MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
      DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.
      DISPLAY "SQLIMSERRMC= " SQLIMSERRMC.
      DISPLAY "SQLIMSSTATE= " SQLIMSSTATE.

************************************************************
*    Display error message
************************************************************
  200-DBERROR.
                                               * See “SQL IMS warning” on page 213
      DISPLAY "ERROR OCCURRED IN WHENEVER SQLWARNING".
      MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
      DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.
      DISPLAY "SQLIMSERRMC= " SQLIMSERRMC.
      DISPLAY "SQLIMSSTATE= " SQLIMSSTATE.
*
************************************************************
*
```

```
************************************************************
 RETURN-TO-CALLER.
- - - - - - - - - - - - - - - - - -   Line(s) not Displayed
                                                  * See "Return to IMS" on page 213
     GOBACK.

 display-sqlda.
                                                  * See "Return to IMS" on page 213
     display "SQLIMSDAID|" SQLIMSdaid
            "|SQLIMSDABC|" SQLIMSdabc
            "|SQLIMSN|" SQLIMSn
            "|SQLIMSD|" SQLIMSd.

     perform with test after
        varying LOOP-COUNT from 1 by 1
        until loop-count > SQLIMSd
        display
          " ",       loop-count,
          " type=",  SQLIMSTYPE(loop-count),
          " len=" ,  SQLIMSLEN(LOOP-COUNT),
          " namel=", SQLIMSNAMEL(loop-count),
          " namec="  SQLIMSNAMEC(LOOP-COUNT),
          "|"
        end-perform.

 END PROGRAM "sqla11".
```

Enterprise COBOL compiler options

The `SQLIMS` compiler option enables the IMS SQL coprocessor, which handles the IMS SQL statements in your program.

IMS SQL Communications Area

Each IMS SQL program must include an IMS SQL Communications Area (IMSSQLCA). This area is included in your application by the compiler, and features several variables that IMS updates after each SQL statement is run.

IMS SQL Descriptor Area

The IMS SQL Descriptor Area (IMSSQLDA) is optional. When it is used with an IMSSQL DESCRIBE statement, it contains a collection of variables that are used to provide information about the columns in the result set. When it is used with an IMSSQL FETCH statement, these fields are used to provide information from your application program to IMS.

Working Storage for IMS SQL data

Your application must provide storage for the information that is provided to and returned from IMS SQL statements. With IMS SQL, information is returned on a column-by-column basis, and the storage that is provided must reflect this. In this area of storage, four items are defined, which are ready for use with IMS SQL statements. The picture clauses of each field must match the definition in the IMS database because no conversion is performed as the fields values are transferred between IMS and your application by the IMS SQL statements.

Linkage section and procedure division statement

This instance is the first time that our application resembles other IMS applications. The requirements for definitions of each PCB in the Linkage Section (and the PCB parameters on the Procedure Division statement) are the same as for any IMS application, whether they use IMS SQL.

Access to IMS continues to be performed through a Program Specification Block (PSB), whether the program uses DL/I calls, IMS SQL statements, or both. The program must define each of the Program Communication Blocks (PCBs) that are defined in the PSB to communicate with IMS. Access to IMS databases continues to use a database PDB (DBPCB).

Whenever statement

The WHENEVER statement specifies the host language statement to be performed when the specified IMS SQL exception condition occurs. Three exceptions can be handled here: SQLERROR, SQLWARNING, and NOT FOUND. Each of these conditions provides for only two options.

The CONTINUE option causes the program to continue on from where the exception occurred to the next statement in the program. The GOTO (or GO TO) option causes control to be passed to the statement that is identified by a `host-label`. In each situation, the IMS SQL Communications Area contains detailed information about the exception condition.

Declare a cursor

The DECLARE CURSOR statement defines a cursor to be used in a subsequent IMS SQL statement. When the associated IMS SQL statement is run, the cursor is used by your program to retrieve the data from the result set that is returned from that IMS SQL statement.

SQL statement text

The SQL statement is stored into a string (PICTURE X) variable in your programs `working-storage`, along with the length of this SQL statement in an associated numeric (PIC 9) variable. The group that contains these two variables (in this case, SQL-STATEMENT) is later passed to IMS SQL to be run.

IMS SQL prepare statement

The IMS SQL PREPARE statement makes the association between the statement text we just set up in our working-storage variable, and the executable form of that statement. In the example that is shown Example 7-2 on page 206, the text was stored in the :SQL-STATEMENT variable. The colon (:) indicates that it is a field or group in our programs working-storage. The DYSQL is the name of the executable form of this statement.

IMS SQL describe output statement

The IMS SQL DESCRIBE OUTPUT statement retrieves information about the prepared SQL statement and stores it in an IMS SQL descriptor area (IMSSQLDA) in our program's working storage. The size of the variable array SQLIMSVAR depends on the value of field SQLIMSn; therefore, we must set this field SQLIMSn to a value that is greater than the number of columns in the result set before we run the describe output statement.

In the statement that is shown in Example 7-3 on page 212, information about the DYSQL prepared statement is retrieved and stored in the SQLIMSda working-storage area in our program.

Example 7-3 SQLIMSDA

```
    EXEC SQLIMS
     INCLUDE SQLIMSDA
    END-EXEC.
01 SQLIMSDA GLOBAL.
  02 SQLIMSDAID         PIC X(8).
  02 SQLIMSDABC         PIC S9(9) COMP-5.
  02 SQLIMSN            PIC S9(4) COMP-5.
  02 SQLIMSD            PIC S9(4) COMP-5.
  02 SQLIMSVAR OCCURS 0 TO 750 TIMES
          DEPENDING ON SQLIMSN.
    03 SQLIMSVAR1.
      04 SQLIMSTYPE     PIC S9(4) COMP-5.
      04 SQLIMSLEN      PIC S9(4) COMP-5.
      04 FILLER REDEFINES SQLIMSLEN.

        05 SQLIMSPRECISION PIC X.
        05 SQLIMSSCALE     PIC X.
      04 SQLIMSDATA     POINTER.
      04 SQLIMSIND      POINTER.
      04 SQLIMSNAME.
        49 SQLIMSNAMEL  PIC S9(4) COMP-5.
        49 SQLIMSNAMEC  PIC X(30).
    03 SQLIMSVAR2 REDEFINES SQLIMSVAR1.

      04 SQLIMSVAR2-RESERVED-1
                        PIC S9(9) COMP-5.
      04 SQLIMSLONGLEN REDEFINES
             SQLIMSVAR2-RESERVED-1
                       PIC S9(9) COMP-5.
      04 SQLIMSVAR2-RESERVED-2
                        PIC S9(9) COMP-5.
      04 SQLIMSDATALEN     POINTER.
      04 SQLIMSDATATYPE-NAME.
        05 SQLIMSDATATYPE-NAMEL
                        PIC S9(4) COMP-5.
       05 SQLIMSDATATYPE-NAMEC PIC X(30).
```

Open the cursor

We are now ready to start using the cursor that we declared earlier (for more information, see "Declare a cursor" on page 211). When the cursor is opened, the DYSQL statement that is associated with the prepared statement (for more information, see "IMS SQL prepare statement" on page 211) is run, and the results are available to the program in the result table.

Close the cursor

When we finish retrieving the rows from the result set, we can close the cursor. Because our program is finished, we return to IMS through the standard **GOBACK** statement.

Fetch the results

Here, we retrieve the next row from the result set with the **FETCH** statement. The cursor **TELE1** is used, and the data is retrieved into the list of data items that were declared in our program's working storage. For the statement that is associated with the **TELE1** cursor, there is only a single column returned, and this is stored in the working storage item `sql-customernumber`. The definition of this item in working storage must match the definition in the database because no data conversion is done with the fetch.

We also check the **SQLIMSCODE** that is returned after the **fetch** call. The program here is rather simplistic; we can also check for other, expected, return codes, including data not found and no more data, and handle these appropriately. The **whenever** statements at the beginning of our program can also be used in these situations.

Data not found

Control is passed to this group of statements when a "data not found" condition is encountered by an IMSSQL statement, as directed by the **whenever** statement at the beginning of our **procedure division**.

SQL IMS error

Control is passed to this group of statements when an "SQL IMS ERROR" condition is encountered by an IMSSQL statement, as directed by the **whenever** statement at the beginning of our **procedure division**.

SQL IMS warning

Control is passed to this group of statements when an "SQL IMS WARNING" condition is encountered by an IMSSQL statement, as directed by the **whenever** statement at the beginning of our **procedure division**.

Return to IMS

When our program finishes, the **goback** statement is used to return to IMS in the usual manner.

Displaying the SQL IMS descriptor area

The **describe output** statement requested a description of the rows to be returned in the result set for our SQL statement earlier in our program. The information is returned in the IMS SQL Descriptor Area (IMSSQLDA). This paragraph displays each of the fields in the IMSSQLDA.

8

IMS Connect enhancements

IMS Connect is a communication gateway that operates in an address space on the z/OS platform between a service consumer (client) and IMS, or IMS and a service producer (server). It interfaces with any TCP/IP supported environment, including Linux.

In this chapter, we describe the IMS Connect enhancements that are available in IMS Version 13.

This chapter includes the following topics:

- XML converter enhancements
- Auto-restart of the Language Environment
- Expanded Recorder Trace records
- Use of RACF Event Notification Facility support for cached RACF UserIDs
- Reporting of overall health to Workload Manager
- Configurable TCP/IP backlog size
- Password phrase support
- Resume TPIPE socket termination enhancement
- Performance enhancements
- IMS Connect command enhancements

8.1 XML converter enhancements

The IMS Connect XML Adapter uses COBOL or PL/I converters to transform XML to byte data and vice versa for the IMS SOAP Gateway functionality. The converters are generated from Rational Developer for System z tooling and loaded in the IMS Connect address space. The IMS application COBOL copybook and Web Service XML schema are used to generate the converters. The IMS Soap Gateway was originally introduced in IMS Version 9 as part of the Integration Suite and is distributed with the IMS Enterprise Suite as the IMS Enterprise Suite SOAP Gateway.

IMS Version 13 includes the following XML converter enhancements:

- A capability to view converters that are loaded

  In IMS Version 12, the WTOR, z/OS Modify, and the Type-2 **UPDATE IMSCON TYPE(CONVERTER)** commands all were used to request a refresh of a converter file that was loaded in IMS Connect. IMS Version 13 includes enhanced the Type-2 command interface with a new command to request detailed converter information.

- Support for extending the current limit of 100 up to 2000 for the maximum number of XML converters.

  The previous maximum of 100 XML Converters that can be loaded at any one time into an IMS Connect instance was increased to 2000. This maximum is controlled through a new MAXCVRT configuration parameter in the ADAPTER statement of the IMS Connect HWSCFGx configuration member of PROCLIB.

  This enhancement is also retrofitted to IMS 12 with the APAR PM64487 (PTF UK79728).

8.1.1 Query support for XML converters

By using the Type-2 command enhancement, SPOC clients that are using the OM API can request detailed information about the converters that are loaded in the IMS Connect instance. If a specific converter is not named, the default for the command is NAME(*).

By requesting TIMESTAMP information, the command output displays the information about this XML Converter’s last successful run (TimeAccess) and the time stamp information when it was last loaded (TimeCreate).

USECOUNT provides the number of times this converter was used since the last time it was loaded.

Example 8-1 shows an example of the new **QUERY IMSCON TYPE(CONVERTER)** command and its output.

Example 8-1 QUERY IMSCON TYPE(CONVERTER) command with its output

```
TSO SPOC input:
QUERY IMSCON TYPE(CONVERTER) NAME(*) SHOW(ALL)

TSO SPOC output:
Converter MbrName CC UseCount TimeAccess          TimeCreate
IMSCVRTD  HWS1     0       30 2011.068 10:50:12.90 2011.067 21:54:11.09
IMSPHBKD  HWS1     0        5 2011.060 21:05:31.78 2011.059 13:55:12.19
```

The output from existing commands that display IMS Connect configuration information, for example (WTOR) VIEWHWS, z/OS MODIFY, and Type-2 QUERY, also are enhanced to also show the current and maximum number of XML converters.

Example 8-2 shows an example of output for existing (WTOR) VIEWHWS, (z/OS MODIFY) F HWS1, and QUERY MEMBER TYPE(IMSCON) SHOW (ALL) commands. The commands were not changed, but the following new fields are displayed:

- MAXCVRT

  The maximum value for the number of XML Converters that can be loaded by this instance of IMS Connect.

- NUMCVRT

  The number of XML Converters that are loaded in this instance of IMS Connect.

Example 8-2 Changed output for VIEWHWS and F HWS1,QUERY commands

```
HWSC0001I   HWS ID=HWS1     RACF=N  PSWDMC=R
HWSC0001I      UIDCACHE=N   UIDAGE=2147483647
HWSC0001I      MAXSOC=50     TIMEOUT=2147483647
HWSC0001I      NUMSOC=5      WARNSOC=55%  WARNINC=5%
HWSC0001I      RRS=Y   STATUS=REGISTERED
HWSC0001I      VERSION=V13 IP-ADDRESS=009.030.114.199
HWSC0001I      SUPER MEMBER NAME=      CMO ACK TOQ=
HWSC0001I    ADAPTER=Y MAXCVRT=400  NUMCVRT=15
HWSC0001I    ODBM AUTO CONNECTION=Y
HWSC0001I    ODBM TIMEOUT=20000
HWSC0001I    ODBM IMSPLEX MEMBER=HWS1             TARGET MEMBER=PLEX1
```

The type-2 command **QUERY IMSCON TYPE(CONFIG)** was enhanced to display the value of the new parameter MAXCVRT in the ADAPTER statement and the number of XML Converters that are loaded in IMS Connect. Two new columns are inserted between the columns Adapter and ODMBAC. This insertion causes the ODBMAC column to shift to the right and any existing columns to the right of it, as shown in Example 8-3.

Example 8-3 An example of QUERY IMSCON TYPE(CONFIG) command with its output

```
QUERY IMSCON TYPE(CONFIG) SHOW(ALL)

TSO SPOC output:  (screen 1)
 MbrName CC  Version IconID IpAddress              MaxSoc TimeOut NumSoc   WarnSoc   WarnInc
 HWS1     0      V13 HWS1   009.030.124.150        50      5000        4        80         5

(screen 2)
 MbrName  UidCache   UidAge           Racf   PswdMc   RRS      RRSStat           Recorder    SMem
CmOAtoq
 HWS1        Y       2147483647       N          N      N      REGISTERED        N           SM01

(screen 3)
 MbrName Adapter MaxCvrt NumCvrt ODBMAC ODBMTO ODBMIpMem ODBMTMem
 HWS1        Y       400      15     Y     18000   IMSPLEX1    PLEX1
```

The SHOW(ADAPTER) is a new option in the **QUERY IMSCON TYPE(CONFIG)** command to display adapter information, including whether XML adapter support is enabled (Y) or disabled (N), the maximum value for the number of XML Converters, and the number of XML Converters that are loaded in this instance of IMS Connect.

Note: Type-2 commands can be accepted only by IMS Connect Version 12 and Version 13 systems. IMS Connect Version 12 systems reject the new type-2 command to query the XML Converters.

8.2 Auto-restart of the Language Environment

IMS Connect uses the Language Environment® and the IMS Base Primitive Environment (BPE) to run the XML Converters when requested by IMS SOAP Gateway. Before IMS Version 13, when one of the XML Converters ends abnormally (ABENDs), the Language Environment instance becomes damaged and further use can result in unpredictable behavior. To restart the Language Environment, IMS Connect must be restarted.

This IMS Version 13 enhancement provides an automated mechanism for restarting the Language Environment after one of the XML Converters ABENDs. This ability means that previously loaded converters are reloaded the next time that they must be used.

IMS Connect also automatically makes a request to BPE to refresh the XMLADAP BPE User Exit after the ABEND limit (ABLIM), which is specified in BPE's EXITDEF() statement for HWS USER EXIT DEFINITION, is reached. By using this auto refresh, IMS Connect users can keep IMS Connect running without having to manually refresh the XMLADAP BPE User Exit for IMS Connect after the ABEND limit is reached.

If the ABLIM specified is a small integer, the XMLADAP user exit is refreshed more frequently during multiple ABENDs. Also, BPE captures a dump for the first ABEND that occurs and for the first ABEND after the XMLADAP BPE User Exit for IMS Connect is refreshed. A value of zero means no ABEND limit.

BPE DISPLAY command can be used to display information about the XML Adapter exit, as shown in Example 8-4.

Example 8-4 An example of BPE DIS USRX command

```
F HWS1,DIS USRX NAME(XMLADAP) OWNER(HWS) SHOW(ABLIM,ABENDS,CALLS)

      Output:
   BPE0030I EXITTYPE MODULE       ABENDS      ABLIM      CALLS
   BPE0000I XMLADAP  HWSXMLA0          0          0          1
   BPE0032I DIS USRX COMMAND COMPLETED
```

8.3 Expanded Recorder Trace records

IMS Connect of IMS 12 introduced new Recorder Trace records that used BPE's External Trace Tables. In IMS Connect of IMS13, these trace records are expanded to capture the entire messages that are sent and received on the DRDA sockets and to and from OM and ODBM via SCI.

Changing the BPE trace level from MEDIUM to HIGH is the trigger to write the enhanced trace records. They also must be written to the external trace data set because they are not written to the fixed size HWSRCDR trace data set. Example 8-5 on page 219 shows the command that is used for activating the enhanced trace.

Example 8-5 Command for activating the writing of enhanced trace records

```
UPDATE TRACETABLE NAME(RCTR) OWNER(HWS) LEVEL(HIGH) EXTERNAL(YES | NO)
```

The macro HWSUSTAT that contains the mapping of the recorder trace records is enhanced and converted to non-Object Code Only (non-OCO). The macro HWSUSTAT is included as a member of the data set IMS.SDFSMAC.

The following mappings are inside the now non-OCO HWSUSTAT macro:

- HWSUSTAT DSECT

  HWSUSTAT DSECT contains the mapping for the short recorder trace records that are produced by IMS Connect. These recorder trace records capture only the first few hundred bytes of the input and output message. These recorder trace records are composed of two sections: an *IPB section for the input message and an *OPB section for the output message.

- HWSLRCTR DSECT

  HWSLRCTR DSECT contains the mapping for the large recorder trace records that are produced by IMS Connect, starting with IMS Connect of IMS 12. The large recorder trace records capture the entire message when it comes into and out of IMS Connect. The large recorder trace records can be differentiated from the short recorder trace records by inspecting the contents of the field _CALLID and by inspecting the contents at the field LRCTR_MSG_TYPE. Also, each of these recorder trace records contain an *IPB or an *OPB section, but not both. Figure 8-1 shows examples of these fields.

```
    Eye ID    MsgType
a- ICONTR.. OTMA ... *IPB  full_message_is_here   <<- TCP/IP receive
b- ICONIS.. OTMA ... *OPB full_message_is_here    <<- XCF send
c- ICONIR.. OTMA ... *IPB  full_message_is_here    <<- XCF receive
d- ICONTS.. OTMA ... *OPB full_message_is_here    <<- TCP/IP send

The other message types are 'ISC ', 'MSC ', 'ODBM', and 'OM  '.
```

Figure 8-1 Eye catchers for enhanced recorder trace records.

As a migration consideration, setting trace level equal to HIGH for Recorder Trace records might produce a significant number of trace records. You should ensure that BPE external tracing is enabled.

8.4 Use of RACF Event Notification Facility support for cached RACF UserIDs

IMS Connect of IMS 12 enabled RACF UserID caching by defining the parameter UIDCACHE=Y in the HWS statement in the HWSCFGx configuration member in PROCLIB, via the type-2 command **UPDATE IMSCON TYPE(CONFIG) SET(UIDCACHE(ON))**, or by using the WTOR command **SETUIDC ON** or the z/OS command **UPDATE MEMBER TYPE(IMSCON) SET(UIDCACHE(ON))**.

These cached user IDs also can be refreshed based on an aging value or manually by issuing the following commands:

- A WTOR command (xx,REFRESH RACFUID)
- A z/OS modify command (**F hws,UPDATE RACFUID NAME..OPTION(REFRESH)**)
- A type-2 command (**UPDATE IMSCON TYPE (RACFUID)**)

The RACF event notification facility (ENF) support for cached UserIDs (UID) enables notification of changes to UIDs that are affected by the **CONNECT**, **REMOVE**, and **ALTUSER REVOKE** RACF commands. IMS Connect is enhanced to listen for the type 71 ENF signals that are produced by these RACF commands and act on that signal to refresh the affected UID. This new capability is applicable only when RACF UID caching is enabled in IMS Connect.

8.5 Reporting of overall health to Workload Manager

Workload Manager (WLM) provides facilities with which servers can indicate their health back to WLM. This health information can be used by the z/OS Sysplex Distributor and considered when connection requests are to be routed to a server.

IMS Connect Version 13 uses the facilities that are provided by WLM to inform WLM of its overall health. As the health of IMS Connect deteriorates or improves, IMS Connect notifies WLM so that WLM has the most recent health information available for use by the Sysplex Distributor. Sysplex Distributor uses the health of each server when the distribution method is set to SERVERWLM in the VIPADISTRIBUTE statement of the TCP/IP profile, as shown in Example 8-6.

Example 8-6 Coding a distribution method of SERVERWLM in TCP/IP PROFILE

```
VIPADYNAMIC …
                VIPADISTRIBUTE DEFINE DISTMETHOD SERVERWLM
```

DISTMETHOD: Incoming connections are routed to multiple stacks that are based on the defined distribution method. The following main options are available:

- ROUNDROBIN
- BASEWLM
- SERVERWLM

For more information, see this website:

http://pic.dhe.ibm.com/infocenter/zos/v1r13/topic/com.ibm.zos.r13.halz001/vpadist.htm

The Sysplex Distributor registers each system with WLM rather than each server on each system. If multiple servers are using the same port on the same system and TCP/IP stack, the health of the system depends on the health of all the servers. Therefore, the connection requests are routed on a per-system health basis first and then on a per-server health basis at each system. This configuration assumes that SHAREPORTWLM is coded in the PORT statement of the TCP/IP PROFLE for the port that is shared by the servers on the same system.

IMS Connect automatically informs WLM of its overall health during initialization and any changes in its health thereafter. The initial health is 100 and the minimum health can be 0. IMS Connect reports its health to WLM whenever the health increases or decreases by at least five percentage points from the last reported value. Also, the reported value is the last multiple of five that is encountered going from the last reported health value to the new health value. This notification algorithm was chosen to reduce the number of times IMS Connect reports its health to WLM, which means that the actual health of IMS Connect at any specific time is within plus or minus five percentage points of the last reported health value. Therefore, if the last reported health is 80%, the health of the IMS Connect at any point can be 76% - 84% without causing an update in the health value reported to WML.

For example, consider the case in which the health is decreasing. Suppose that the last reported health is 80%. Now, suppose that three actions occur that reduce the health of the IMS Connect by approximately 16%. Suppose two of the activities bring the health down by 10%. Because the reporting threshold is 5% and 10% is larger than or equal to 5%, IMS Connect reports its new health of 70% to WLM.

The last activity occurs and the health is reduced by the last 6%, which brings the health down to 64%. Because 6% is larger than or equal to 5%, IMS Connect reports a change in health to WLM. The health is reported as 65% and not 64% because 65% is the last multiple of 5 that was encountered going down from 70% to 64%.

The opposite scenario where the health is increasing is as follows: if the last reported health is 50% and it increases to 64%, IMS Connect reports its health as 60%. The health is reported as 60% and not as 64% because 60% is the last multiple of 5 that was encountered going up from 50% to 64%.

The Sysplex Distributor and the TCP/IP stack can use the health value to determine which IMS Connect is the most viable for routing connection requests. Consider the following points:

- WLM assigns a weight ranging 1 - 64. WLM can start a weight adjustment in favor of one system to help workloads that are not meeting their goals. SERVERWLM weights are based on a comparison of target servers within the same service class; for example, how well the server is meeting the goals of its service class and how much displaceable capacity is available on this system for new work that is based on the importance of the service class. The target systems poll WLM for their server weights each minute and forward the weights to the distributor. When all systems are running at or near 100% utilization, WLM assigns higher weights to the systems with higher amounts of lower importance work; for example, those with the most displaceable capacity.
- If there are multiple IMS Connects on each system that coded the SHAREPORTWLM parameter, the connection requests are divided among the IMS Connects that are based on their health. Each system's health can be a maximum of 100%. Two IMS Connects at 100% each on a single system equals 100%.

Note: WLM does not consider the health of IMS Connect before IMS Version 13 when determining the weight to assign to each IMS Connect. Therefore, earlier releases of IMS Connect can receive more connection requests in a mixed environment that is compared to IMS Connects at IMS Version 13 level.

8.5.1 Single system examples

Suppose System 1 has two IMS Connects: HWS1 and HWS2. Both IMS Connects can process the same transactions and therefore, are using the same port, PORT 9999. To allow the TCP/IP stack to consider the health of each IMS Connect before a connection request is sent, the PORT statement in the TCP/IP PROFILE must be modified to use SHAREPORTWLM for PORT 9999. Figure 8-2 shows the single System 1 example.

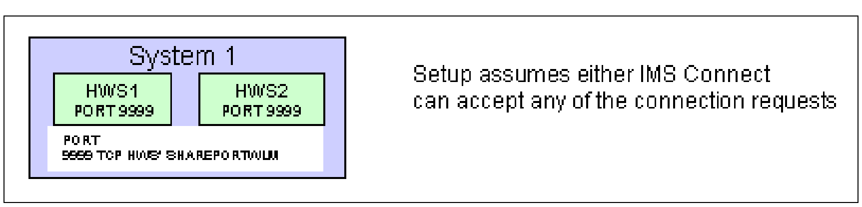


Figure 8-2 Single System 1 example

8.5.2 Sysplex Distributor dual system example

Suppose System 1 has two IMS Connects: HWS1 and HWS2, and System 2 has one IMS Connect (HWS3). All three IMS Connects can process the same transactions and therefore, are using the same port, PORT 9999.

So that the TCP/IP stack can consider the health of each IMS Connect at each system before a connection request is sent, the PORT statement in the TCP/IP PROFILE must be modified to use SHAREPORTWLM for PORT 9999. Figure 8-3 shows a Sysplex Distributor (SD) (System 1 and System 2) example.

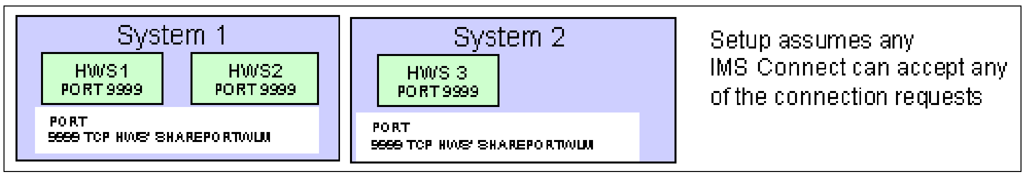


Figure 8-3 Sysplex Distributor (SD) (System 1 and System 2) example

Figure 8-4 shows sample outputs for the **DISPLAY TCPIP** command for a system that is using SERVERWLM versus BASEWLM versus ROUNDROBIN. You can see the health of the System for the specified port when SERVERWLM is used. With BASEWLM or ROUNDROBIN, the health of the system is not reported or considered by the DISPLAY command.

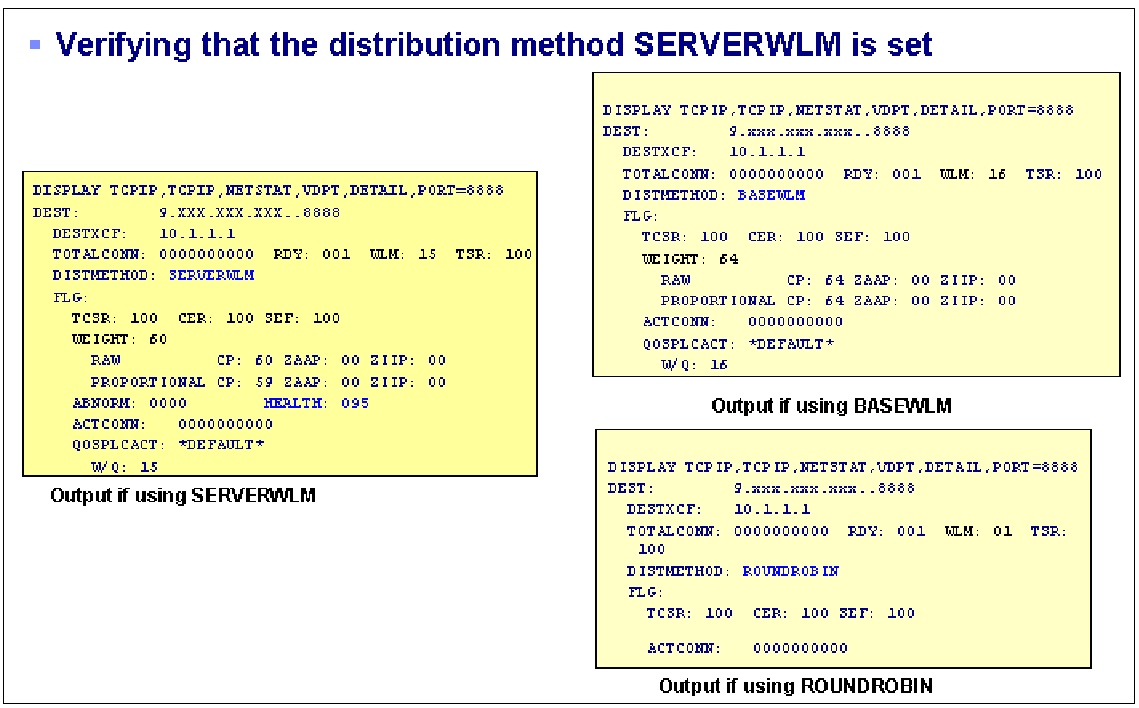


Figure 8-4 Sample outputs for the DISPLAY TCPIP command

8.6 Configurable TCP/IP backlog size

Before IMS 13, IMS Connect establishes a TCP/IP listening socket on every port that is configured for its use when IMS Connect starts. When these listening sockets are established, IMS Connect also sets the corresponding TCP/IP backlog or queue value to be equal to the MAXSOC value for IMS Connect. If not specified, the default for MAXSOC is 50. This backlog or queue is the number of connection requests that can be queued in TCP/IP while they are waiting for IMS Connect to assign sockets to them; for example, during peak processing times. This configuration means that requests up to this value can be queued in TCP/IP without their connections being rejected pending IMS Connect action. If IMS Connect's value is higher than that specified for SOMAXCONN in the TCP/IP profile, the smaller of the two values (SOMAXCONN or MAXSOC) is used by the TCP/IP stack.

IMS Connect V13 adds a new parameter to the configuration member for IMS Connect. This new parameter supports a value that can be different from MAXSOC for the TCP/IP backlog. However, the same rules regarding SOMAXCONN in the TCP/IP profile continue to apply. If TCPIPQ is greater than SOMAXCONN, the number that is used is the one that is defined for SOMAXCONN. Therefore, the new TCPIPQ parameter value must be coordinated with the value in the SOMAXCONN. For more information, see *z/OS Communications Server: IP Configuration Reference,* SC27-3651.

The TCPIPQ parameter is defined in the TCP/IP statement in HWSCFGxx as shown in the following example:

```
HWSCFGxx :  TCPIP...  TCPIPQ=nnnnnnnnnn
```

TCPIPQ can be set to a maximum of 2147483647 (X'7FFFFFFF'). IMS Connect uses a default value of 50, which was the minimum value that can be set in previous releases of IMS Connect. If the value is 0 - 49, IMS Connect defaults to 50. If the value is less than zero or larger than 2147483647, IMS Connect ABENDs with a message that indicates the value is outside the legal range by issuing the existing HWSX0909E and BPE0003E error messages.

8.7 Password phrase support

IMS Connect in IMS Version 13 supports security password phrases. You can use a phrase with maximum of 100 characters instead of password (maximum of 8 characters). This enhancement is provided with IMS TM Resource Adapter enhancements and is described in 2.4, "RACF password phrase support" on page 37.

8.8 Resume TPIPE socket termination enhancement

IMS Connect in IMS Version 13 provides a socket enhancement for Resume option Wait-For-Ever session (listening socket). With this enhancement, the IMS Connect session that is in CONN state (waiting on reply from datastore) with execution timeout value of "wait-for-ever" as a result of a Resume TPIPE 'auto' or 'single' wait immediately detects a session termination of the client application.

This function is provided by APAR PM90777 (PTF UK95578). For more information, see 3.1.6, "OTMA Callable Interface (OTMA C/I) asynchronous enhancement" on page 77.

8.9 Performance enhancements

IMS Connect in IMS Version 13 provides performance enhancements in the following areas:

- Use of CPOOL instead of STORAGE OBTAIN in XCF message exit:
  - Needed for performance
  - Adds 167 MB of Extended Private to the IMS Connect address space
  - Plans to increase the REGION size or size that is allowed by the IEFUSI exit
- Use of zIIP where appropriate and available:
  - DRDA threads
  - SOAP Gateway threads
  - MSC over TCP/IP threads
  - ISC over TCP/IP threads

For more information about IMS 13 and IMS Connect performance enhancements, see Chapter 9, “IMS performance enhancements” on page 231.

8.10 IMS Connect command enhancements

IMS 13 provides new dynamic capabilities for entering IMS Connect commands. IMS Connect users can now add new ports and IMS datastore connections without having to restart IMS Connect by using the new type-2 commands for IMS Connect.

This enhancement provides higher availability by eliminating the need to restart IMS Connect to activate definitional changes. The new type-2 CREATE commands are targeted for all IMS Connect customers. The CREATE commands are intended to be used to dynamically create definitions that might not be known or needed during the creation of the static definitions.

Common Service Layer is a requirement for this enhancement as for any type-2 commands with a minimum of Structured Call Interface (SCI) and Operations Manager (OM) address spaces set up.

This section includes the following topics:

- CREATE IMSCON TYPE(PORT) command
- CREATE IMSCON TYPE(DATASTORE) command
- Operational considerations
- Security considerations

8.10.1 CREATE IMSCON TYPE(PORT) command

You can now dynamically create an IMS Connect port by using the type-2 **`CREATE IMSCON TYPE(PORT)`** command. The use of this command is equivalent to defining TCPIP (regular port) or ODACCESS (DRDA port) statements in HWSCFGxx.

Example 8-7 shows the syntax for **`CREATE IMSCON`** command.

Example 8-7 Syntax for CREATE IMSCON command

```
CREATE IMSCON TYPE(PORT)
        NAME(portnum1, portnum2, ...)
        LIKE(portnum_model)
        SET(attribute1, attribute2, ...)
```

The command includes the following parameters:

- NAME(portnum)

  Specifies the port number.

- LIKE(portnum_model)

  Allows an existing port to be used as a model and is optional.

- SET(attribute)

  Explicitly defines settings (optional).

  PORTTYPE(REG|DRDA) specifies a regular or DRDA type of port:

  – "Regular" is equivalent to defining PORT substatement in TCPIP statement TCPIP=(PORT=(ID=, KEEPAV=, EDIT=),...). REG is the default.

  – "DRDA" is equivalent to defining DRDAPORT substatement in ODACCESS statement ODACCESS=(DRDAPORT=(ID=, KEEPAV=, PORTTMOT=),...)

By using the SET( ) parameters, a user can define a keep alive value for a port. For a regular port, the Port Edit Exit routine can also be specified here; for a DRDA port, the timeout value can be set.

The optional SET attribute can be used for explicitly defining the following settings:

- KEEPAV()

  Sets the "keep alive" value and defaults to 0 (the TCP/IP stack's setting is used).

- EDITRTN()

  Specifies a port edit exit routine and is valid only for regular ports.

- PORTTMOT()

  Sets the timeout value for a DRDA port. Default is 6000 hundredths of a second (1 minute).

Figure 8-5 on page 226 shows examples of the **CREATE IMSCON TYPE(PORT)** command and the equivalent control statement in HWSCFGxx. In the example that shows the dynamic creation of a regular port, the SET(PORTTYPE()) parameter was omitted because the default setting is a regular port type.

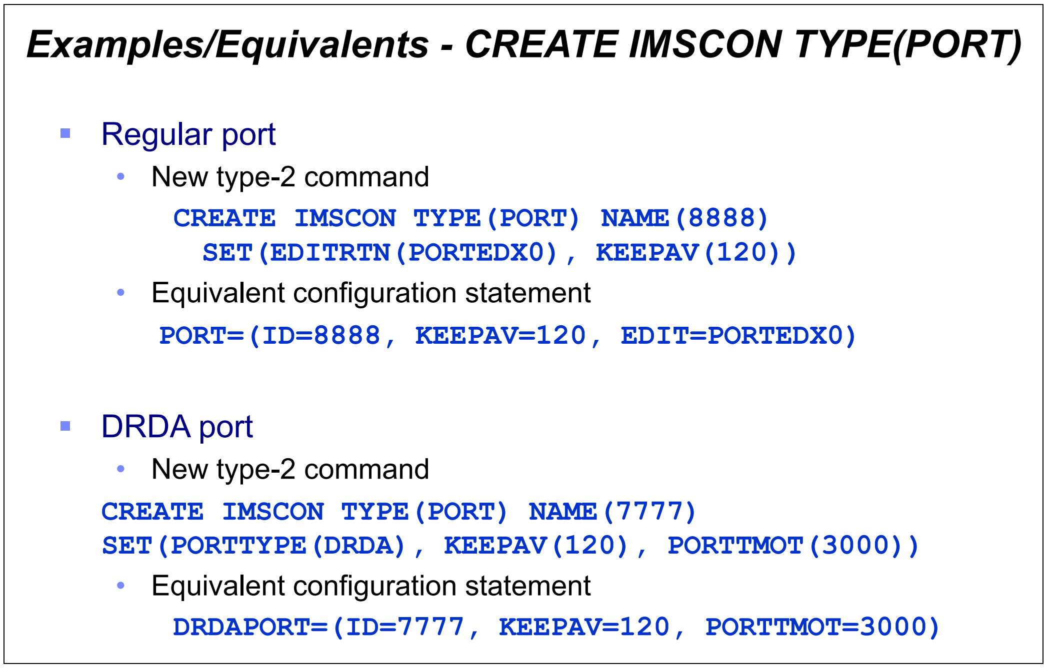


Figure 8-5 Examples of CREATE IMSCON commands with equivalent control statements

Figure 8-6 shows an example of the use of the type-2 **CREATE IMSCON** command to dynamically create a port. The commands are asynchronous and a completion code (CC) of 0 shown on the TSO SPOC output window that indicates that the command was submitted to IMS Connect. When the port is successfully dynamically created, an HWSS0790I message is output to the console. The M=SDOT that is shown represents the module code that issued the message (M= is included in all IMS Connect messages).

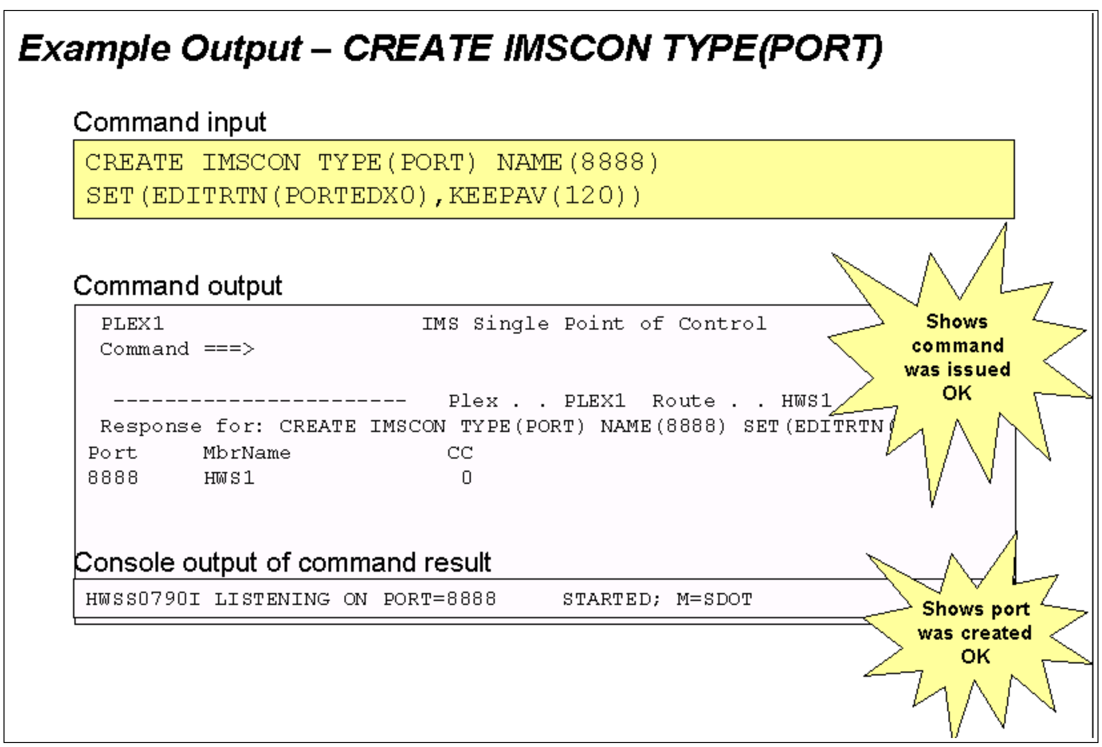


Figure 8-6 CREATE IMSCON TYPE(PORT) output example

8.10.2 CREATE IMSCON TYPE(DATASTORE) command

You can now dynamically create an IMS Connect datastore by using the type-2 **CREATE IMSCON TYPE(DATASTORE)** command. This command is equivalent to defining the DATASTORE statement in HWSCFGxx. Example 8-8 shows the syntax for **CREATE IMSCON TYPE(DATASTORE)** command.

Example 8-8 Syntax for CREATE IMSCON TYPE(DATASTORE) command

```
CREATE IMSCON TYPE(DATASTORE)
        NAME(datastore_name)
        LIKE(datastore_model)
        SET(attribute1, attribute2, ...
```

The command includes the following parameters:

- NAME

  Required parameter that specifies the 1 - 8 character name of the IMS datastore. The name can consist of alphanumeric characters and must be unique within IMS Connect. This parameter is equivalent to the ID parameter of the DATASTORE statement in the IMS Connect configuration member.

- LIKE

  Specifies the existing IMS Connect DATASTORE resource to use as a model. The new resource is created with all the same attributes as the model. Attributes set explicitly by the CREATE command overrides the model attributes. Later changes to the model are not propagated to resources that were created from it.

- SET

  Specifies the attributes of the IMS datastore to be created. If the LIKE keyword is omitted, the required attributes must be specified and the optional attributes take their default values. If the LIKE keyword is specified, the attributes that are specified override the model attributes. For the SET( ) parameter, the only value that is required is for MEMBER( ). If a LIKE( ) parameter is not specified, values for GROUP( ) and TMEMBER( ) are also required. The rest of the values for SET( ) have defaults.

  The SET parameter includes the following options:

  - SET(ACKTO(nnn))

    Specifies a timeout interval for acknowledgements to OTMA for CM0 and CM1 output messages and for IMS to IMS transaction messages. The timeout value can be 0 - 255 seconds. This parameter is optional and defaults to 120. If the timeout value is 0 or is not specified, the OTMA ACK timeout default value of 120 seconds is set. For IMS to IMS transaction messages, if an acknowledgment is not received by OTMA before the timeout interval expires, OTMA reroutes the transaction message to the timeout queue DFS$$TOQ.

  - SET(APPL(applname))

    Specifies a 1 - 8 character alphanumeric TCP/IP APPL name that is defined to RACF in the PTKTDATA statement. This parameter is optional and defaults to blanks. If you are using PassTicket and user message exits, you must specify this APPL parameter.

  - SET(CM0ATOQ())

    Specifies name of the OTMA CM0 ACK timeout queue and overrides the OTMA default value of DFS$$TOQ and any value set on the HWS statement of the IMS Connect configuration member.

- SET(DRU(druname))

  Specifies name of the OTMA destination resolution user (DRU) exit that is passed to OTMA. Defaults to DFSYDRU0.
- SET(GROUP(grpname))

  Specifies a name of the z/OS cross-system coupling facility group for the IMS OTMA.
- SET(MAXI(nnnn))

  Specifies the OTMA input message flood control value. The valid range is 0 - 9999. If you do not specify a value or specify a value of 0, the OTMA default value of 5000 is used. If you specify a value of 1 - 200, the OTMA minimum value of 200 is used. This parameter is optional and defaults to 5000.
- SET(MEMBER(mbrname))

  Specifies an XCF member name that identifies IMS Connect in the XCF group that is specified by the GROUP parameter. Required.
- SET(OAAV(nnnn)

  Defines the OTMA Access Control Environment Element (ACEE) aging value, in seconds, for this IMS data store. When reached, OTMA refreshes the ACEE before it processes the next input message that is received from IMS Connect. Valid values are 0 - 999999. If you specify 0, OTMA uses the default value of 999999. If you specify a value 1 - 300, OTMA uses a value of 300 seconds.
- SET(RRNAME(altdest))

  Specifies the name of the alternative destination of a client's reroute request. The name can consist of alphanumeric characters (A - Z, 0 - 9) and special characters (@, #, and $). IMS Connect translates lowercase characters to uppercase characters. This parameter is optional and defaults to HWS$DEF.
- SET(SMEMBER(smbrname))

  Specifies the name of the OTMA super member to which this IMS data store belongs. If specified, this value overrides the value of the SMEMBER parameter that is specified on the HWS statement in the IMS Connect configuration member. To disable the value of SMEMBER that is specified on this IMS data store, specify the parameter with no value; for example, SMEMBER(). This parameter is optional and defaults to blanks.
- SET(TMEMBER): Specifies the XCF member name of the IMS that this IMS Connect communicates within the XCF group.

Figure 8-7 on page 229 shows examples of the **CREATE IMSCON TYPE(DATASTORE)** command and the equivalent control statement in HWSCFGxx. The example is for dynamically creating a DATASTORE by using the type-2 CREATE IMSCON command; however, this time it references the name of an existing datastore to set the attribute values.

The TMEMBER( ) parameter setting is optional, but in this case it was appropriate to explicitly specify it to override what is set otherwise. If TMEMBER(IMS3) was not specified here, the default is TMEMBER(IMS1) because that setting is associated with the IMS1 datastore that is referenced with LIKE(IMS1).

Examples/Equivalents - CREATE IMSCON TYPE(DATASTORE)

- Example using the LIKE keyword referencing an exisitng Datastore definition
  - New type-2 command

```
CREATE IMSCON TYPE(DATASTORE) NAME(IMS3) LIKE(IMS1)
  SET(MEMBER(ICON3), TMEMBER(IMS3))
```

  - Equivalent configuration statement

```
DATASTORE=(ID=IMS3, MEMBER=ICON3, TMEMBER=IMS3,
  APPL=APPL1, ACTO=120, CM0ATOQ=TOQ1, DRU=DFSYDRU0,
  GROUP=XCFGRP1, MAXI=5000, OAAV=7200, RRNAME=RR1)
```

Figure 8-7 Examples of CREATE IMSCON commands with equivalent control statements

Figure 8-8 shows an example of the use of the type-2 **CREATE IMSCON** command to dynamically create a datastore. The commands are asynchronous and a completion code (CC) of 0 that is shown on the TSO SPOC output window indicates that the command was submitted to IMS Connect. When the datastore is successfully dynamically created, an HWSS0790I message is output to the console. The M=DSC1 shown represents the module code that issued the message (M= is included in all IMS Connect messages).

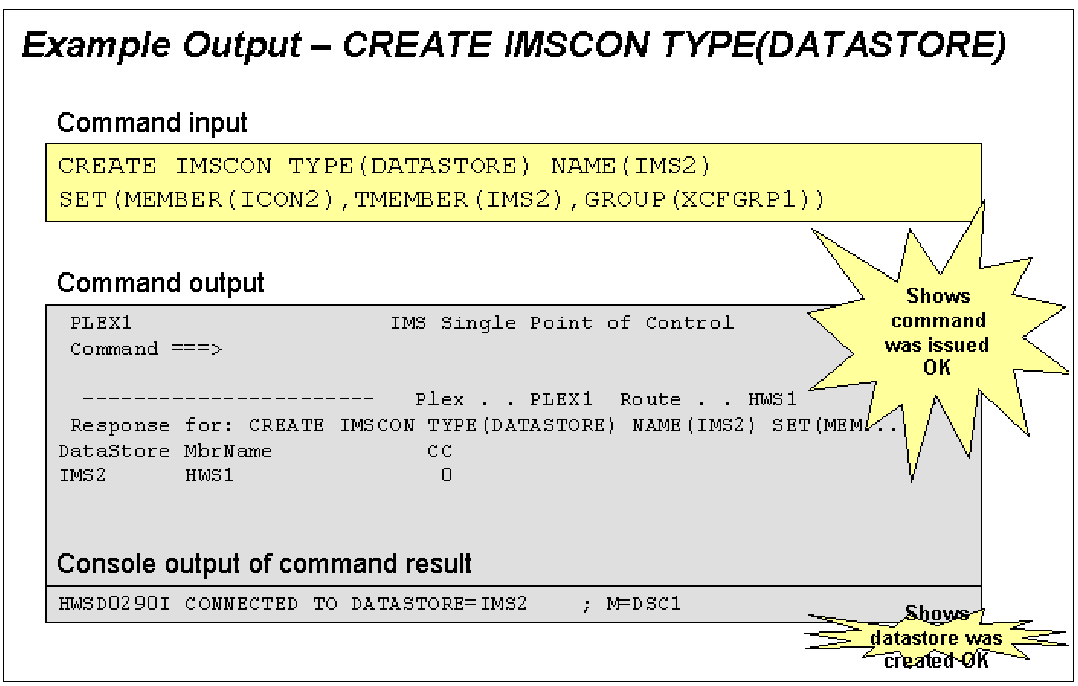


Figure 8-8 Example of CREATE IMSCON TYPE(DATASTORE) -output

In the TSO **SPOC** command input example, the command was routed to the IMS Connect that is named HWS1. If all IMS Connects in the IMSplex are sharing the datastore, the user can create the datastore definition for each IMS Connect by leaving the Route field blank so that each IMS Connect receives the command. In the TSO **SPOC** output that is shown, the datastore is dynamically created for the HWS1 IMS Connect.

8.10.3 Operational considerations

Definitions that are created by **CREATE** commands do not persist across IMS Connect restart. To preserve newly created ports and datastores across restarts of IMS Connect, make sure that the following definitions in HWSCFGxx are updated:

- For dynamically created regular ports, update the PORT subparameter of the TCPIP statement.
- For dynamically created DRDA ports, update the DRDAPORT subparameter of the ODACCESS statement.
- For dynamically created datastores, update the DATASTORE statement.

8.10.4 Security considerations

Add RACF definitions for OM in OPERCMDS class to restrict access to the **CREATE IMSCON** command. Table 8-1 shows the RACF attributes that you can use for securing the **CREATE** command.

Table 8-1 RACF attributes for CREATE command

IMS command	Keyword	RACF access authority	Resource name
CREATE	IMSCON	UPDATE	IMS.plxname.CRE.IMSCON

9

IMS performance enhancements

IMS is the high-performance application and data server for IBM System z. IMS 13 continues to build on its tradition of high-performance, low-cost transaction processing by removing constraints and reducing path length while using new system functions. IMS 13 offers a combination of performance, scalability, usability, and reliability for customer's mission-critical applications.

This chapter includes a description of some of the specific, largest contributors to the performance enhancements for IMS 13 and is largely based on the Silicon Valley Lab performance measurements that are documented in the white paper *IMS 13 Performance Evaluation Summary - Reducing the Total Cost of Ownership with Improved Performance*, which is available at this website:

`http://ibm.co/1jy804r`

As such, the standard performance notices apply.

> **Note:** The information that is contained in this document was submitted to any formal IBM test and is distributed on an "AS IS" basis without any warranty that is expressed or implied. 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 their operational environment. Although each item might be reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results can be obtained elsewhere. Customers that are attempting to adapt these techniques to their own environments do so at their own risk.

Note: Performance is based on measurements and projections that use IMS benchmarks in a controlled environment. The results vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, I/O configuration, storage configuration, amount of System z Integrated Information Processors (zIIP) capacity that is available during processing, and the workload processed. Therefore, results can vary significantly and no assurance can be given that an individual user can achieve results that are similar to those stated here. Results should be used for reference purposes only.

The test scenarios (hardware configuration and workloads) that are used in this document to generate performance data are not considered "best performance case" scenarios. Performance might be better or worse depending on the hardware configuration, data set types and sizes, and the overall workload on the system.

Note: The information that is provided in this chapter was obtained at the IBM Silicon Valley Laboratory and is intended for migration and capacity planning purposes.

This chapter includes the following topics:

- Performance overview
- Log latch rewrite enhancement
- Block serialization latch enhancement
- Universal database driver type-2 enhancements
- IMS Connect/OTMA enhancements
- zIIP performance considerations
- Improved SVC Directory Tables dump formatting
- Performance enhancements conclusion

9.1 Performance overview

IMS continues to demonstrate that it can provide the highest performance, lowest cost transaction performance with absolute integrity for messages and database. IMS 13 also provides several enhancements to reduce internal IMS path length, which can reduce overhead and cost per transaction.

These enhancements are spread throughout the product where IMS can find improved methods to accomplish the same function more efficiently.

The latest release makes IMS more efficient by improving CPU efficiency, which reduces elapsed time and contention that allows better throughput scaling. This is achieved by using new hardware features and new software algorithms that improve performance and reduce contention and real-time delays, which leads to the lower cost per IMS transaction.

9.1.1 IMS 13 performance enhancements

IMS 13 includes new enhancements and functionality that can potentially reduce IMS total cost of ownership (TCO) for customers through improved performance.

The performance improvements over IMS 12 also allow IMS 13 to reach an unprecedented milestone in high volume transaction processing by using Fast Path with a single IMS image.

In addition to the reduced TCO in IMS 13, various new functions and enhancements were added in all of its components.

9.1.2 Performance improve changes

IMS 13 performance improvements are not limited only to high throughput scenarios; they provide reduced contention and path length, optimized storage allocation, and improved Fast Path/OTMA sync point processing at all transaction volume levels.

IMS 13 improves performance and reduces TCO of IMS from its beginning through many changes. The following changes affect the workloads:

- Contention reduction (specifically, log latch reduction and several other internal IMS latches and local lock requests).
- Path length reduction through improved algorithms and the elimination of inefficient or expensive code and instructions, especially in the OTMA and Fast Path areas.
- IMS Universal Database driver type-2 optimization.
- Batch processing contention and elapsed time reduction through improvements in the Block Serialization latch management.
- More use of zIIPs.

9.1.3 Some IBM lab measurements

IBM lab measurements of IMS 13 show the following enhancements:

- Multiple concurrently executing BMP jobs that are performing many inserts and deletes complete in up to 30% lower elapsed time that is compared with IMS 12.
- A log latch contention reduction of up to 80% compared with IMS 12. This contention reduction can improve your overall throughput and cost per transaction.

Table 9-1 shows some basic metrics that apply to almost all workload measurements.

Table 9-1 Basic metrics in workload measurements

Metric	Description
CPU busy%	The average percent busy across all general CPUs on an LPAR during the measurement interval.
CPU service time	The amount of CPU time that was used by the processors on an LPAR. This can also be reported on a per transaction basis by dividing the total CPU service time that is used during an interval by the number of transactions that are processed in that interval.
ETR	External Transaction Rate, which is the observed average transaction rate in transactions per second (TPS) over the measurement interval.
ITR	Internal Transaction Rate, which is the ETR divided by CPU busy%. ITR is a projection of the observed transaction rate (ETR) to what the transaction rate is if the processor were running at 100% CPU busy, assuming linear scaling.
ITR delta%	Higher delta is good and indicates that the workload is more efficient and the overall service time per transaction is lower.

9.2 Log latch rewrite enhancement

The IMS log latch rewrite enhancement is part of the IMS 13 enhancements for reduced TCO. It can reduce log latch contention and CPU time that is used by the logger, particularly at high logging rates, which can improve transaction throughput while using fewer resources.

This enhancement allows multiple tasks to log records in parallel. Before IMS 13, the logger serialized the process of copying log records in to a buffer by getting an internal “latch” called the log latch (LOGL) in exclusive mode. This meant that only one caller at a time moved a log record in to a buffer.

In IMS 13, the new algorithm for log buffer space assignment serializes exclusively only for the length of a single instruction (“compare-and-swap”) at the following points:

- Buffer space assignment
- WADS process synchronization
- Record copy completion

Although contention can still occur at these points, it is resolved by retrying the compare-and-swap. No waiting or suspension of the caller is required.

9.2.1 Log latch still in exclusive mode

The log latch can still be obtained in exclusive mode by logger administrative processing (such as OLDS switching) and by callers that are outside of the logger.

For this reason, logger space assignment still obtains the log latch in shared mode to serialize with these still-exclusive latch owners. Therefore, waits for the LOGL latch in the logger still can occur; however, their frequency is reduced.

9.2.2 IMS log buffer management redesigned

Also, as a part of this enhancement, the IMS log buffer management processing was redesigned. This change reduces CPU usage in the logger for installations by using more log buffers, which also eliminates transient waits for log buffers that can occur with the older versions of IMS.

9.2.3 The LREC latch

Additionally, a new latch (called the LREC latch) was introduced so that IMS programs outside of the logger can use it for serializing the building of log records.

This latch serializes with other requesters of the LREC latch, but not with general callers of the IMS logger for Fast Path.

The Fast Path sync point was changed to obtain this new latch instead of the LOGL latch when building and logging its log records.

Although this change does not eliminate all contention (LREC requesters might still have to wait for other callers that are holding the LREC latch), it avoids penalizing logger callers who are logging unrelated log records that do not require such tight serialization.

9.2.4 Evaluation results of the log latch rewrite enhancement

The Fast Path and Full Function workloads were used to evaluate the Log Latch Rewrite enhancement.

Full Function workload

IMS 13 showed a significant reduction in log latch contention over IMS 12 for the Full Function workload. Performance results demonstrate 66% reduction in total log latch contention per second. Also, ETR and ITR increased and CPU usage decreased.

Figure 9-1 shows the comparison between IMS 12 and IMS 13.

Log Latch Rewrite: Full Function Workload				
	IMS 12	**IMS 13**	**Delta**	**Delta %**
ETR (Tran/Sec)	8,188.62	9,014.54	825.92	10.09%
CPU Busy%	86.76%	82.44%	-4.32	-4.98%
ITR	9,438.24	10,934.67	1,496.43	15.85%
Total Log Latch Contention (LOGL/sec) (Sum of Exclusive and Shared)	13,904	4,649.82	9,254.18	-66.56%
Log Records/sec	267,400	294,828	27,428	10.26%

Figure 9-1 IMS 13 log latch rewrite Full Function evaluation results

Fast Path workload

IMS 13 showed a significant reduction in log latch contention over IMS 12 for the Fast Path workload. Performance results demonstrate 88% reduction in total log latch contention. Also, ETR and ITR increased and CPU usage decreased.

Figure 9-2 shows the comparison between IMS 12 and IMS 13.

Log Latch Rewrite: Fast Path Workload				
	IMS 12	IMS 13	Delta	Delta %
ETR(Tran/Sec)	73,391	77,595	4,204	5.73%
CPU Busy%	84.23%	83.16%	-1.07%	-1.27%
ITR	87,132	93,308	6,176	7.09%
Log Latch Contention (LOGL/sec)	72,647	46	-72,601	-99.94%
Log Record Latch Contention (LREC/sec)	0	8,121	8,121	--
Total Latch Contention	72,647	8167	64480	-88.75%

Figure 9-2 IMS 13 log latch rewrite Fast Path evaluation results

The higher contention reduction was observed in Fast Path workload because of the introduction of new LREC latch that is specifically for Fast Path building of log records.

It leads to the drastic reduction of log latch contention from thousands in IMS 12 to a double-digit value in IMS 13.

9.2.5 Summary

IMS log buffer management processing was designed to realize the following benefits:

- Potentially reduce CPU usage in the logger for installations that use many log buffers
- Significantly reduce transient waits for log buffers

For Fast Path-specific processing, anew latch (called the LREC latch) was introduced so that IMS programs outside of the logger can use it for serializing the building of log records.

Full Function and Fast Path workload was used to evaluate the performance. This system enhancement benefits to customers with high IMS log volume.

Evaluation summary

IMS 13 log latch rewrite enhancement can significantly reduce log latch contention and CPU usage in Fast Path and Full Function environments.

Figure 9-3 on page 237 shows the total log latch contention reduction in IMS 13 from IMS 12.

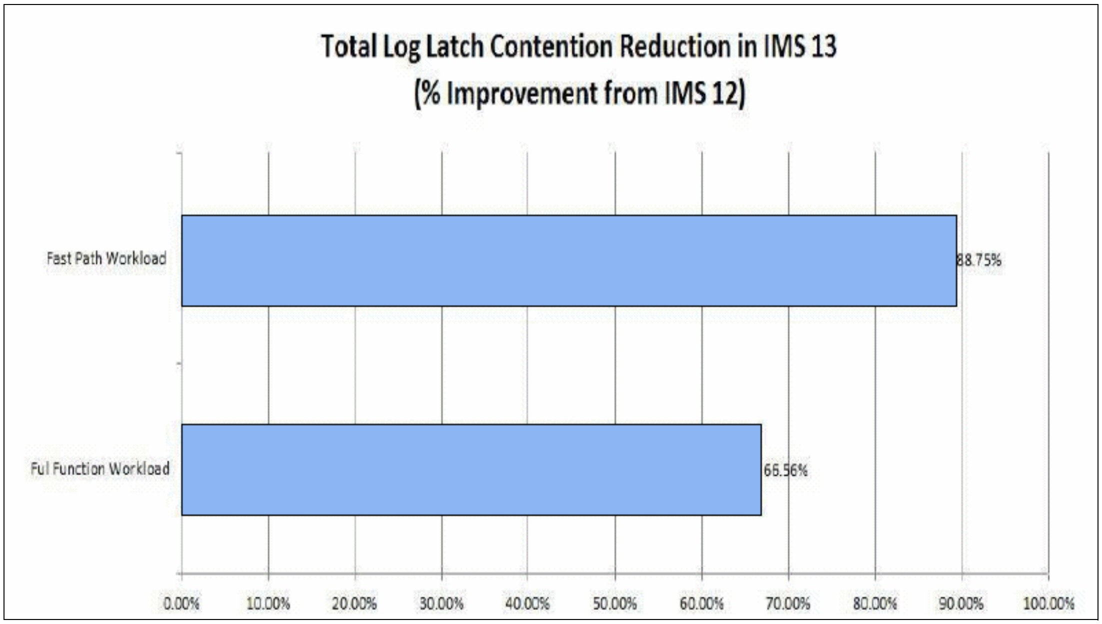


Figure 9-3 Total log latch contention reduction in IMS 13 from IMS 12

9.3 Block serialization latch enhancement

Database block serialization is a method that is used by IMS space management process to protect unused space in HD type databases, including HALDBs. Consider the following points:

- All OSAM blocks and VSAM ESDS (not KSDS) control intervals have free space elements that are used to manage unused space.
- As segments are inserted or deleted, the free space chain is updated by space management to reflect what part of the space is unused.

Serialization is needed to protect the integrity of the free space chains.

To avoid the overhead of the IMS locking protocols, space management uses special latches that are called *block serialization* to provide serialization.

The elements are a chain of latch headers and a main latch that is used to serialize the chain.

9.3.1 Overview

IMS DB Block Serialization uses an internal chain of latches to ensure integrity when segments are inserted into or deleted from an IMS Full Function database.

The elements are a chain of latch headers. The chain of latches is protected by a main latch header that is used to serialize the chain.

This enhancement reduces internal latch contention, which can result in lower elapsed times and reduced CPU cost for heavy batch (BMP) workloads.

9.3.2 Before IMS 13

Before IMS 13, there was only one block serialization latch header across all of IMS that can result in significant contention and increased elapsed times during heavy insert and delete activity.

This configuration presents an IMS bottleneck because every task across IMS that is trying to get or free space in HD databases is funneled through a single latch header.

Latch waiters are chained in LIFO order. It is possible for an IMS task on a busy system to wait for the main latch for a long time before getting it if many other tasks wait on the main latch before the earliest waiter gets the main latch.

This issue can lead to an overall slowdown of IMS if the earlier waiters hold other IMS resources, such as locks or other latches.

9.3.3 In IMS 13

In IMS 13, the number of latch headers was increased from 1 to 1001 based on the data management block (DMB) number, which significantly reduces the likelihood of contention for this latch.

The DB block serialization latch was split from a single latch to multiple latches. Normally, this latch is not a problem; however, it can be an issue when there is heavy BMP activity.

The reduction in contention can lead to shorter elapsed times and allow for more concurrency, especially when multiple concurrent BMP jobs with a large amount of insert and delete activity are run.

9.3.4 Evaluation results of the block serialization latch enhancement

IMS 13 measurements show approximately 50% reduction in elapsed time with highly concurrent workload when compared to IMS 12 that is running the same workload with heavy insert and delete activity.

IMS 13 showed a significant reduction in elapsed time and CPU time (task and SRB) over IMS 12 for the highly concurrent BMP workload. A significant CPU cost saving also was observed; roughly 11% in total CPU task time.

Customers with similar workloads might observe a similar reduction in the elapsed time for BMP batch processing.

Figure 9-4 on page 239 shows the comparison between IMS 12 and IMS 13.

Block Serialization Enhancement: Full Function Non-HALDB Workload				
	IMS 12	IMS 13	Delta	Delta %
Total Elapsed time spent to execute all 20 BMPs (sec)	181	89	92	-50.83%
Average BMP Elapsed time spent for all 20 BMPs (sec)	178.22	87.71	90.51	-50.79%
Total CPU Task Time (sec)	451.43	399.75	51.68	-11.45%
Total SRB Time (sec)	11.05	1.14	9.91	-89.68%

Figure 9-4 IMS 13 block serialization enhancement evaluation results

9.3.5 New statistics in IMS log record

In addition to the performance aspects of this enhancement, there are new statistics that are externalized in the x'4507' log record that can be used to monitor the block serialization latch.

9.3.6 Summary

This performance enhancement benefit customers who are running multiple concurrently executing BMP jobs that are performing many inserts and deletes.

Evaluation summary

Figure 9-5 shows the evaluation summary of the block serialization latch enhancement in IMS 13 from IMS 12.

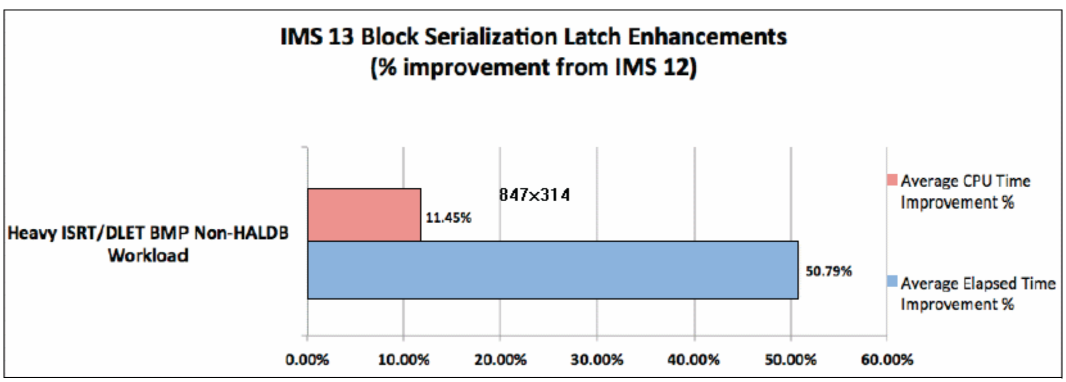


Figure 9-5 Elapsed time and CPU time Comparison with IMS 12 and IMS 13

9.4 Universal database driver type-2 enhancements

The IMS Transaction Manager evolved to provide data that is easily accessible and shareable so that customers can derive more value from past investments, control costs, and optimize their IT infrastructure.

9.4.1 Overview

The type-2 connectivity of the IMS Universal Database drivers (IMSUDB) provides local, non-TCP/IP access to IMS databases when the IMS Universal Database driver application is running locally on the same LPAR as IMS subsystem by using IMS Open Database Access (ODBA) and IMS database resource adapter (DRA) interfaces.

Type2 connectivity is supported from the applications that are running on the following systems:

- IMS Dependent Regions (JBP, JMP, MPP, IFP, BMP)
- WebSphere Application Server for System z
- CICS for z/OS

The focus of this enhancement was to reduce the elapsed time and the amount of CPU that is used when IMS Universal Database drivers with type-2 connectivity are run.

9.4.2 Enhanced IMS 13 performance characteristics

The most enhanced areas include the batch queries (for example, SQL SELECT with large fetch sizes), SQL DELETE, and SQL UPDATE with large set of matches for update and delete.

Type-2 connectivity uses mix of Java and native language calls to access IMS data. Internally, it calls Java Native Interface (JNI) to access lower-level C interfaces to get to IMS.

This enhancement focuses on reducing the amount of time spent to cross the JNI layer. The IMS Java Batch Processing (JBP) workload was used to test this enhancement

9.4.3 JBP workload environment

A single LPAR was configured to conduct the IMS Universal driver type-2 connectivity enhancement evaluation by using JBP workload with the IMS 13 Universal driver to compare against the IMS 12 Universal driver.

The JBP workload application is also based on the Fast Path workload and runs an IMS Universal driver application, which then drives IMS DL/I calls to IMS.

The following applications were run to measure enhancements:

- A SELECT application was run with fetchSize=1 to compare single row fetch between releases of the IMSUDB drivers. The application returns 13437500 rows of data.
- A SELECT application was run with fetchSize=20000 to compare batch and multirow fetch between releases of the IMSUDB drivers. The application returns 13437500 rows of data.
- A DELETE application deletes 311350 rows of data to compare batch delete between releases of the IMSUDB drivers.
- A UPDATE application updates 311350 rows of data to compare batch update between releases of the IMSUDB drivers.

9.4.4 Evaluation results of IMSUDB type-2 enhancements

In this section, we examine the following types of SQL statements:

- SELECT with fetchSize=1
- SELECT with fetchSize=20000
- SQL DELETE (batch delete)
- SQL UPDATE (batch update)

SELECT with fetchSize=1

IMS 13 showed a reduction in CPU task time that was used over IMS 12 by the batch application that used SELECT with fetchSize=1 to retrieve 13437500 records.

Additionally, IMS 13 showed a reduction in elapsed time (wall clock time) and a negligible reduction in CPU SRB time.

Figure 9-6 shows the comparison between IMS 12 and IMS 13.

Select Fetch Size 1 (13,437,500 records retrieved)	IMS 12 UDB Driver	IMS 13 UDB Driver	Delta	Delta %
CPU Task Time (seconds)	78.39	64.19	-14.20	-18.11%
CPU SRB Time (seconds)	4.9	4.88	-0.02	-0.41%
Region Used Above 16M (K)	301,840	335,064	33,224	11.01%
Elapsed Time (seconds)	266	251	-15.00	-5.64%

Figure 9-6 JBP workload SELECT with fetchsize=1 evaluation results

SELECT with fetchSize=20000

IMS 13 showed a reduction in CPU task time that was used over IMS 12 by the batch application that used SELECT with fetchSize=20000 to retrieve 13437500 records.

Additionally, IMS 13 showed a reduction in elapsed time (wall clock time) and CPU SRB time.

Figure 9-7 shows the comparison between IMS 12 and IMS 13.

Select Fetch Size 20,000 (13,437,500 records retrieved)	IMS 12 UDB Driver	IMS 13 UDB Driver	Delta	Delta %
CPU Task Time (seconds)	85.45	40.86	-44.59	-52.18%
CPU SRB Time (seconds)	4.9	4.65	-0.25	-5.10%
Region Used Above 16M (K)	301,136	325,016	23,880	7.93%
Elapsed Time (seconds)	268	222	-46.00	-17.16%

Figure 9-7 JBP workload SELECT with fetchsize=20000 evaluation results

SQL DELETE (batch delete)

IMS 13 showed a reduction in CPU task time that was used over IMS 12 by the batch application that uses SQL DELETE (batch delete) to delete 311,350 records.

Additionally, IMS 13 showed a reduction in elapsed time (wall clock time) and CPU SRB time.

Figure 9-8 on page 242 shows the comparison between IMS 12 and IMS 13.

Batch Delete (311,350 records deleted)	IMS 12 UDB Driver	IMS 13 UDB Driver	Delta	Delta %
CPU Task Time (seconds)	6.1	3.03	-3.07	-50.33%
CPU SRB Time (seconds)	1.5	1.41	-0.09	-6.00%
Region Used Above 16M (K)	284,652	274,404	-10,248	-3.60%
Elapsed Time (seconds)	64	61	-3.00	-4.69%

Figure 9-8 JBP workload SQL DELETE evaluation results

SQL UPDATE (batch update)

IMS 13 showed a reduction in CPU Task time that was used over IMS 12 by the batch application that used SQL UPDATE (batch update) to update ~3 million records.

Additionally, IMS 13 showed a reduction in elapsed time (wall clock time) and CPU SRB time.

Figure 9-9 shows the comparison between IMS 12 and IMS 13.

Batch Update (3,113,500 records updated)	IMS 12 UDB Driver	13 IMS UDB Driver	Delta	Delta %
CPU Task Time (seconds)	31.16	11.8	-19.36	-62.13%
CPU SRB Time (seconds)	1.16	1.08	-0.08	-6.90%
Region Used Above 16M (K)	314,628	278,688	-35,940	-11.42%
Elapsed Time (seconds)	93	78	-15.00	-16.13%

Figure 9-9 JBP workload SQL UPDATE evaluation results

9.4.5 Summary

IMS 13 Universal Database driver that uses type-2 connectivity demonstrates reductions in CPU time and in elapsed time that is compared to IMS 12.

Evaluation summary

Table 9-2 on page 243 lists the IMS Java batch workload improvements that were observed after the evaluation of the IMS 13 Universal Database driver type-2 enhancements.

Table 9-2 IMS Java batch workload improvements

Improvements	Retrieve	Delete	Update
CPU task time	52%	50%	62%
Elapsed time	17%	4%	16%

These improvements are shown in Figure 9-10.

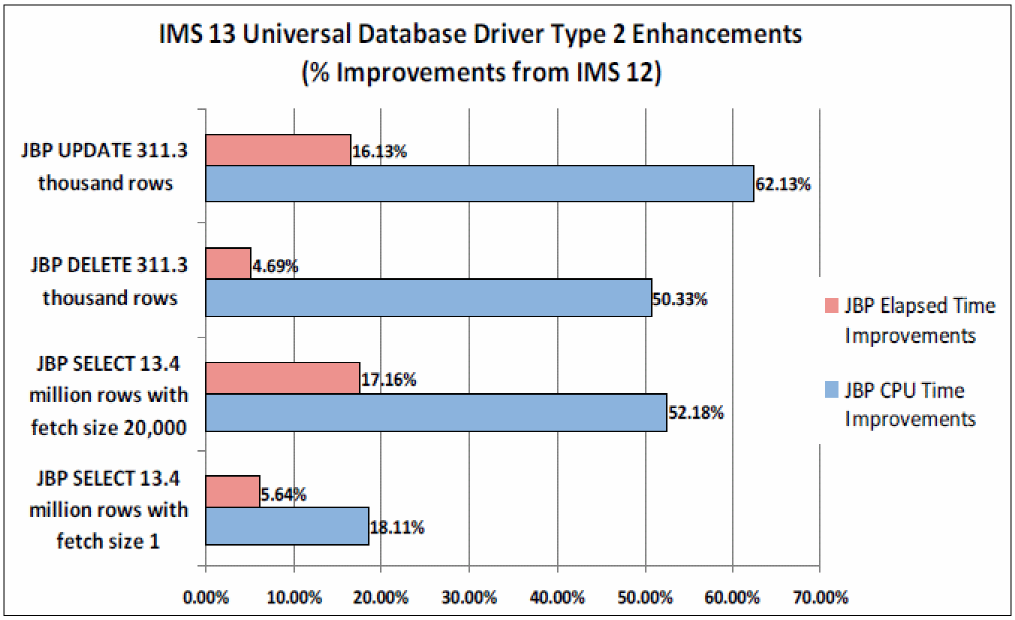


Figure 9-10 IMS 13 Universal Database driver type-2% improvements from IMS 12

9.5 IMS Connect/OTMA enhancements

In IMS 13, the following enhancements were made in the IMS Connect and OTMA area that are related to performance:

- OTMA client processing

  OTMA YTIB chain was changed to improve FINDDEST performance.

- Storage management

  Conversion of OTMA and IMS Connect z/OS STORAGE calls to CPOOL.

- Buffer cleanup

  Removal of unnecessary clearing of OTMA buffers.

- Syncpoint

  Fast optimized Fast Path/OTMA sync point processing.

9.5.1 OTMA client processing

In IMS 13, an internal OTMA data structure was changed from a singly linked list to a hash table. This change improves the performance for searching for an ITMA control block that represents the source of and is associated to an input transaction message.

OTMA YTIB chain changed from a single linked list to a hash table to improve FINDDEST performance. With a hash table, the length of the chain that must be scanned is greatly reduced. Processor cache contention is also reduced because fewer blocks must be referenced by a processor for any specific search.

The effect on a specific IMS environment depends on its use of OTMA and the volume of activity. For example, every CM1 (Send-then-commit) transaction request has a control block that is associated with the port TPIPE during processing.

If a slowdown in IMS or z/OS occurs or if the system is actively processing large numbers of requests, an increasing number of these blocks remain active.

The improved hashing technique results in a more timely and efficient search for a specific block during application GU or checkpoint.

9.5.2 Storage management

IMS Connect and OTMA were changed to more efficiently obtain and release fixed-sized storage areas in XCF input exits.

z/OS STORAGE OBTAIN and RELEASE calls in OTMA and IMS Connect for these areas were replaced with more efficient storage management service calls (CPOOL and DFSBCB).

The cell pool macro (CPOOL) provides programs with another way of obtaining virtual storage other than the STORAGE OBTAIN macro.

CPOOL provides centralized, high-performance cell management services. DFSBCB (BCB storage management interface macro) is an IMS storage manager service that manages storage in specific storage pools. This change can reduce contention on the address space local lock and path length that is required to get and free the storage.

9.5.3 Buffer cleanup

OTMA output processing for synchronous (CM1) transactions always obtained and cleared a large (maximum output-sized) buffer.

This clearing was not necessary, except for sync level 2 (SYNCPOINT) messages.

In IMS 13, this unneeded buffer clear was removed for sync level NONE and CONFIRM.

9.5.4 Syncpoint

The Fast Path/OTMA sync point processing was changed such that sync level NONE and COMMIT local (that is, not shared EMH) messages are processed similar to SNA messages.

This enhancement removed a check write (CHKW) call under the dependent region, and instead issues the logger write call for the sync point record so that a notification is sent to the IMS control region when that record is hardened to disk.

The OTMA send-deallocate is issued by the control region when that notification is received.

The ITASK that processes this notification runs under a new Fast Path TCB (TCB type FP2) to minimize any interference with the existing XFP Fast Path TCB processing.

This approach frees the dependent region sooner, so that it can be available to process the next transaction rather than waiting for logger I/O.

9.5.5 Evaluation of the IMS Connect/OTMA enhancements

IMS Connect/OTMA enhancements were tested by using Fast Path and Full Function workload that is driven by many TCP/IP clients.

Full Function workload

In this section, we describe the IMS Connect/OTMA enhancements for Full Function workload that is driven by many TCP/IP clients.

IMS 13 showed an improvement in ETR and ITR at similar CPU usage over IMS 12 for the Full Function/high number of clients workload.

IMS lab measurements showed an 8% improvement in ITR when comparing IMS 13 to IMS 12 for a Full Function workload. The following enhancements contributed to this improvement:

- Log latch contention reduction
- IMS Connect/OTMA enhancements
- OTMA find destination call serialization

Figure 9-11 shows the comparison between IMS 12 and IMS 13.

IMS Connect/OTMA Enhancements Performance Results (Full Function workload)				
	IMS 12	IMS 13	Delta	Delta %
ETR(Tran/Sec)	4,623	4,894	271	5.86%
CPU BUSY %	80.94%	79.04%	-1.90%	-2.35%
ITR	5,711	6,192	481	8.42%

Figure 9-11 Full Function: IMS 13 IMS Connect/OTMA enhancements evaluation results

The results also show that as the number of TCP/IP clients increase, the ITR also increases.

Fast Path workload

In this section, we describe the IMS Connect/OTMA enhancements for Fast Path workload that is driven by many TCP/IP clients.

IMS 13 showed a significant improvement in ETR and ITR at similar CPU usage over IMS 12 for the Fast Path/high number of clients workload.

IMS lab measurements showed a 163% improvement in ITR when comparing IMS 13 to IMS 12 for a Full Function / IMS Connect workload.

The improvement in ITR was the result of implementing the following enhancements in the Fast Path area:

- Log latch contention reduction
- IMS Connect/OTMA enhancements
- OTMA find destination call serialization
- Optimized sync point processing

Figure 9-12 shows the comparison between IMS 12 and IMS 13.

IMS Connect/OTMA Enhancements Performance Results (Fast Path workload)				
	IMS 12	IMS 13	Delta	Delta %
ETR(Tran/Sec)	8814	23270	14456	164.01%
CPU Busy %	78.68%	78.72%	0.04%	0.05%
ITR	11202	29560	18358	163.88%

Figure 9-12 Fast Path: IMS 13 IMS Connect/OTMA enhancements evaluation results

9.5.6 Summary

IMS 13 OTMA and IMS Connect processing provide greater efficiency and scalability for many OTMA/IMS Connect clients in Full Function and Fast Path.

The specific IMS 12 to IMS 13 comparisons in Figure 9-13 show an extreme case in which a many IMS Connect clients were used to drive an IMS Full Function and an IMS Fast Path. This workload was used to highlight the effects of the IMS Connect and OTMA contention reduction improvements.

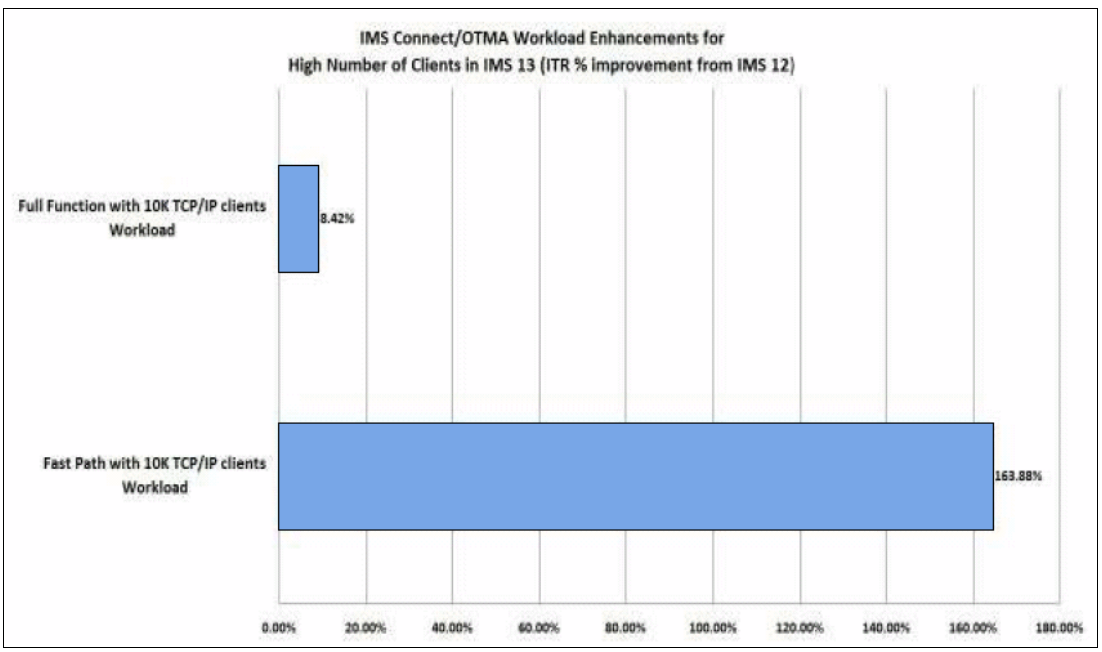


Figure 9-13 IMS 13 IMS Connect/OTMA enhancements evaluation comparison from IMS 12

These comparisons are extreme corner cases and should not be used to set expectations about any particular customer environment.

The comparisons also are more of a demonstration of the internal contention reduction and transaction rate scalability achievable with IMS 13.

9.6 zIIP performance considerations

Note: This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (SEs); for example, zIIPs, zAAPs, and IFLs. IBM authorizes customers to use IBM SE to run only the processing of Eligible Workloads of specific programs that are expressly authorized by IBM as specified in the Authorized Use Table (AUT) for IBM Machines, which are available at this website:

http://www.ibm.com/systems/support/machine_warranties/machine_code/aut.html

No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors or Central Processors because customers are authorized to use SEs to process only certain types or amounts of workloads as specified by IBM in the AUT.

We described the use of zIIP by IMS in 2.8, "IMS zIIP utilization enhancements" on page 58. The goal of zIIP offloading is to reduce the amount of general processor time that is used to perform a task by moving some of this time to zIIPs.

In IMS 13, certain processing in the IMS Connect address space and in the IMS ODBM address space can be enabled by the customer to use zIIPs, if they are available. Enabling zIIP usage might result in reduced general CPU consumption, depending on the customer's workload characteristics.

It is possible that performance might improve if the general processors were close to maximum usage. It is also possible that doing so increases the total amount of processing time (that is, the sum of general CP time + zIIP time).

For example, consider the case of a thread that performs some sort of processing that includes issuing a supervisor call (SVC) and calling a user exit (neither is allowed in SRB mode), as shown in Figure 9-14.

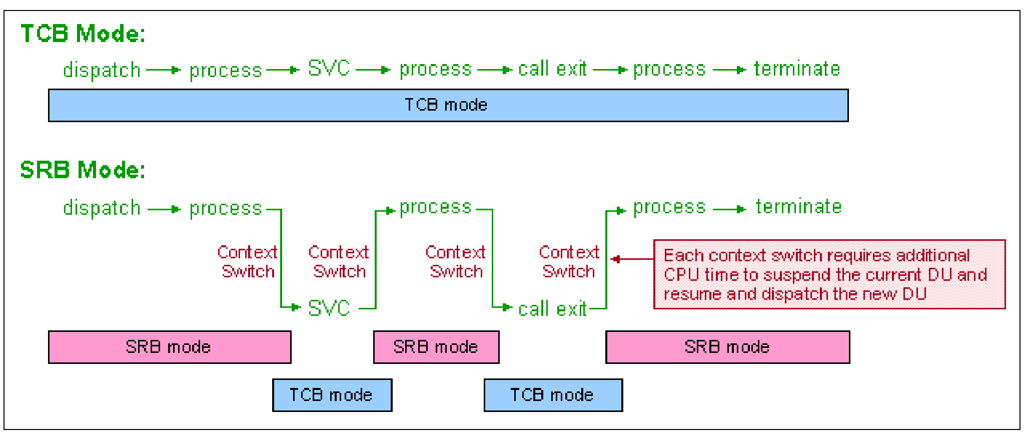


Figure 9-14 Thread that performs a process, issues a SAN Volume Controller, and calls a user exit

The following performance trade offs can occur:

- If zIIPs are used, the general CPU consumption might be reduced; however, the total CPU (general + zIIP) might increase for the same amount of work.
- If zIIPs are not used, the CPU used might be higher, but the total CPU might be less.

Heavy use of user exits can affect the amount of general CPU reduction that is seen when it is running with zIIPs. This affect occurs because user exits must be driven in TCB mode. Each call to a user exit requires a context switch, from SRB mode to TCB mode and back (see Figure 9-14 on page 247). Results are customer-specific and based on workload, the amount of user exit processing, and so on.

9.6.1 DRDA workload zIIP utilization

This section describes the performance characteristics of zIIP usage for a DRDA Workload.

For Open Database DRDA workloads, threads that process the DRDA requests in IMS Connect and ODBM are zIIP eligible.

Figure 9-15 show the IMS Open Database DRDA flow.

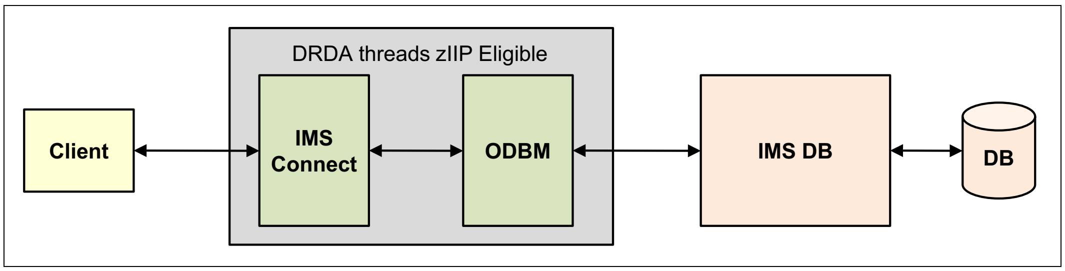


Figure 9-15 IMS open database DRDA flow

Environments

Several performance tests were run with zIIP usage enabled [CONDSRB(ALWAYS)] and disabled [CONDSRB(NEVER)] in the BPE configuration PROCLIB members for the IMS Connect and ODBM address spaces. The environment consisted of an IMS Universal Database Driver for Java test driver application that is running on an LPAR with five general processors (GPs) that are connected via TCP/IP to another LPAR with five GPs and two zIIPs on a z196 system.

The tests were done with two different ODBM environments that used the following Resource Recovery Service (RRS) enabled (RRS=Y) and disabled (RRS=N):

- `RRS=N`

  In this environment, RRS is not used and ODBM uses the IMS DRA (DBResource Adapter).

- `RRS=Y`

  In this environment, RRS is used and ODBM uses IMS ODBA (Open DBAccess).

  The DRDA Workload drives various IMS DL/I calls on many parallel threads and portions of the DRDA calls that were made can run on zIIP.

- The average number of DL/I calls per IMS schedule was 1.75.

The following number of DRDA client threads that were used for the workload varied for the RRS=Y and RRS=N measurements:

- RRS=Y used 25 DRDA client threads
- RRS=N used 35 DRDA client threads

Consumption

The total CPU time on the GPs and zIIPs for each measured interval was divided by the number of transactions that were processed to give an average number of CPU microseconds that are used per transaction (GP and zIIP) to normalize the comparisons.

The General and zIIP CPU microseconds that are used per transaction was calculated by multiplying the number of General or Specialty Purpose Engines, General or zIIP CPU% Busy, 1000000 and dividing by the transaction rate, as shown in the following examples:

```
General CPU µs/tran = (No. of General Purpose Engines * General CPU % Busy
* 1000000)/ETR (Tran/Sec)

zIIP CPU µs/tran = (No. of Specialty Purpose Engines * zIIP CPU % Busy *
1000000)/ ETR(Tran/Sec)
```

The percentage comparison values that were reported are percentage reductions in GP CPU microseconds per transaction between the baseline (zIIP usage that was disabled for the IMS Connect and ODBM address spaces, as appropriate for the workload being measured) and the zIIP usage that was enabled.

> **Note:** All workloads that are described in this section were measured in an environment with enough CPU capacity (GPs and zIIPs) to run without contending for CPU cycles.

Evaluation results of the DRDA workload zIIP utilization

The DRDA workload results varied with RRS=Y and RRS=N processing.

RRS=Y processing

RRS=Y measurements demonstrate the following results:

- A 15.21% savings in GP (General Purpose Engine) microseconds that were used per transaction when they were running with zIIP utilization enabled, compared with running without zIIP utilization enabled.
- A 7.34% increase in total CPU (GP + zIIP) microseconds that were used per transaction when they were running with zIIP utilization enabled, compared with running without zIIP utilization enabled.

Figure 9-16 on page 250 shows the DRDA workload result with RRS=Y.

zIIP Utilization Evaluation for DRDA Workload using RRS=Y				
	zIIP Disabled	**zIIP Enabled**	**Delta**	**Delta %**
ETR (Tran/Sec)	4165.45	4272.55	107.1	2.57%
General CPU Busy %	83.71%	72.80%	-10.91	-13.03%
zIIP Busy %	5.05%	53.98%	48.93	968.91%
ITR	4976.05	5868.88	892.83	17.94%
General CPU µs/Tran	1004.81	851.95	-152.86	-15.21%
zIIP CPU µs/Tran	24.25	252.68	228.44	942.12%
Total CPU µs/Tran	1029.06	1104.63	75.57	7.34%

Figure 9-16 DRDA workload that uses RRS=Y evaluation results

RRS=N processing

RRS=N measurements demonstrate the following results:

- A 22.77% savings in GP (General Purpose Engine) microseconds that were used per transaction when they were running with zIIP utilization enabled, compared with running without zIIP utilization enabled.
- A 14.01% increase in total CPU (GP + zIIP) microseconds that were used per transaction when they were running with zIIP utilization enabled, compared with running without zIIP utilization enabled.

Figure 9-17 shows the DRDA workload result with RRS=N.

zIIP Utilization Evaluation for DRDA Workload using RRS=N				
	zIIP Disabled	**zIIP Enabled**	**Delta**	**Delta %**
ETR (Tran/Sec)	6892.03	6239.68	-652.35	-9.47%
General CPU Busy %	70.35%	49.19%	-21.16%	-30.07%
zIIP Busy %	4.81%	63.52%	58.71	1220.58%
ITR	9796.77	12684.85	2888.08	29.47%
General CPU µs/Tran	510.37	394.17	-116.20	-22.77%
zIIP CPU µs/Tran	13.96	203.60	189.64	1358.65%
Total CPU µs/Tran	524.33	597.77	73.44	14.01%

Figure 9-17 DRDA workload that use RRS=N evaluation results

Summary

From a performance perspective, when using zIIP enhancements, there is a trade-off for customers between the reduced GP usage (lower cost) and more total CPU usage on both GPs and zIIPs, which leads to longer total path length.

9.6.2 SOAP workload zIIP utilization

This section describes the performance characteristics of zIIP utilization for a SOAP Workload.

Threads that process the SOAP requests in IMS Connect are zIIP eligible.

Figure 9-18 show the IMS SOAP gateway flow.

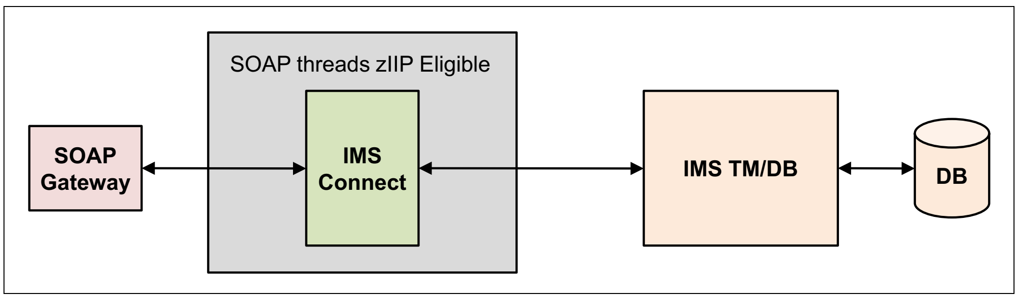


Figure 9-18 IMS SOAP gateway flow

Environment

A performance test was run with zIIP utilization enabled [CONDSRB(ALWAYS)] and disabled [CONDSRB(NEVER)] in the BPE configuration PROCLIB members for the IMS Connect address space. The environment consisted on IMS Enterprise Suite SOAP Gateway Driver application running on an LPAR with 4 GPs connected via TCP/IP to another LPAR with 2 GPs and 2 zIIPs on a z196 system.

The SOAP workload drives transactions that receive an input message and return an output message. The transactions receive an input message size of ~1,931 bytes and an output message size of ~9,073 bytes.

Consumption

Portions of the IMS Connect handling of SOAP message threads can run on zIIP.

The test driver-used 300 SOAP client threads to drive the workload.

Evaluation results of the SOAP zIIP Utilization

The lab environment resulted in the following SOAP Workload measurements:

- An 8.47% savings in GP microseconds that were used per transaction when it was running with zIIP utilization enabled, compared with running without zIIP utilization enabled.
- A 0.31% reduction[1] in total CPU (GP + zIIP) microseconds that were used per transaction when it was running with zIIP utilization enabled, compared with running without zIIP utilization. Figure 9-19 on page 252 shows the SOAP workload results.

[1] The reduction in total CPU (GP + zIIP) is within measurement variability. The expectation is that, when solely running a similar SOAP workload, the total CPU typically shows a negligible increase from the added logic that is needed to make processing available to the zIIPs.

zIIP Utilization Evaluation for SOAP Workload				
	zIIP Disabled	zIIP Enabled	Delta	Delta %
ETR (Tran/Sec)	1428.73	1499.1	70.37	4.93%
General CPU Busy %	65.31%	62.72%	-2.59%	-3.96%
zIIP Busy %	3.55%	9.31%	5.76%	162.25%
ITR	2187.61	2390.14	202.53	9.25%
General CPU µs/Tran	914.23	836.76	-77.46	-8.47%
zIIP CPU µs/Tran	49.69	124.20	74.51	149.92%
Total CPU µs/Tran	963.93	960.97	-2.95	-0.31%

Figure 9-19 SOAP workload evaluation results

9.6.3 MSC and ISC workload zIIP types

The following examples of zIIP utilization for existing workload types are using new facilities:

- MSC over TCP/IP (IMS Connect)

  IMS to IMS Communication (new in IMS 12)
- ISC over TCP/IP (IMS Connect)

  IMS to CICS Communication (new in IMS 13)

MSC over TCP/IP (IMS Connect) flow

Figure 9-20 show the flow for MSC over TCP/IP workloads with new IMS 12 function. Threads processing the MSC requests in IMS Connect are zIIP eligible.

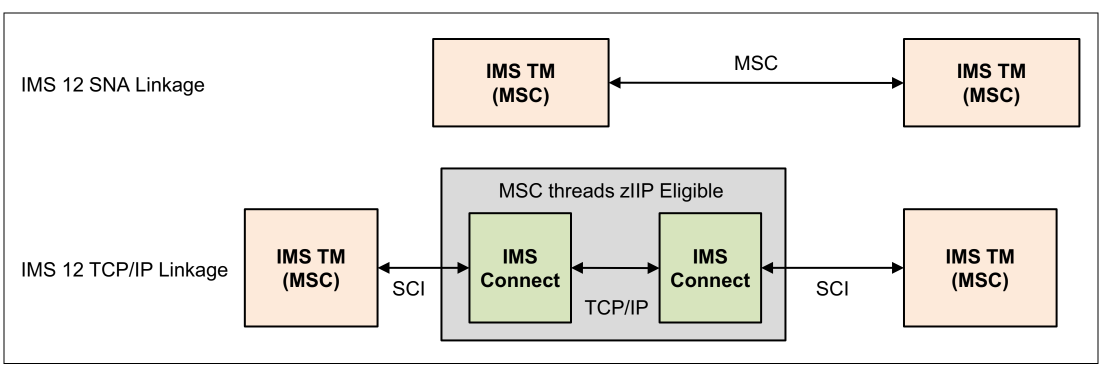


Figure 9-20 Flow for MSC over TCP/IP workloads

ISC over TCP/IP (IMS Connect) flow

Figure 9-21 on page 253 shows the flow for ISC over TCP/IP workloads with new IMS 13 function. Threads processing the ISC requests in IMS Connect are zIIP eligible.

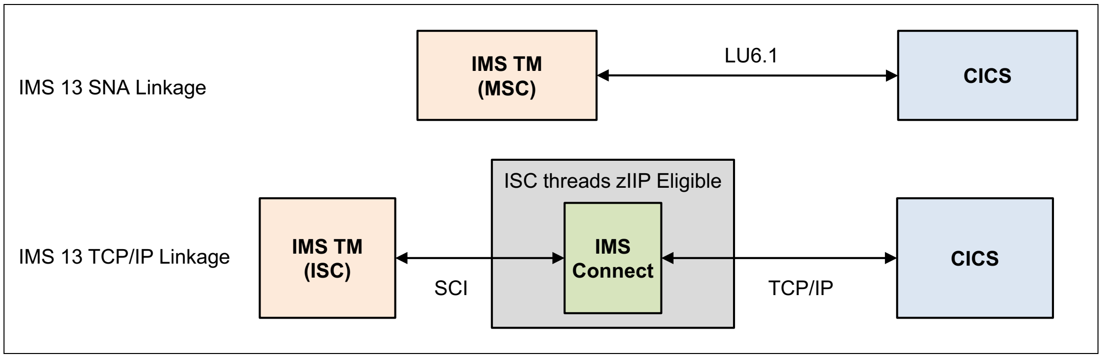


Figure 9-21 Flow for ISC over TCP/IP workloads

9.6.4 MSC workload zIIP utilization

This section describes the performance characteristics of zIIP utilization MSC workload.

MSC customers who are planning to move from SNA to TCP/IP with this function might benefit from this enhancement.

For MSC over TCP/IP workloads, threads that are processing the MSC requests in IMS Connect are zIIP eligible.

Environment

A performance test was run with zIIP utilization enabled [CONDSRB(ALWAYS)] and disabled [CONDSRB(NEVER)] in the BPE configuration PROCLIB members for the IMS Connect address space. The environment is shown in Figure 9-22.

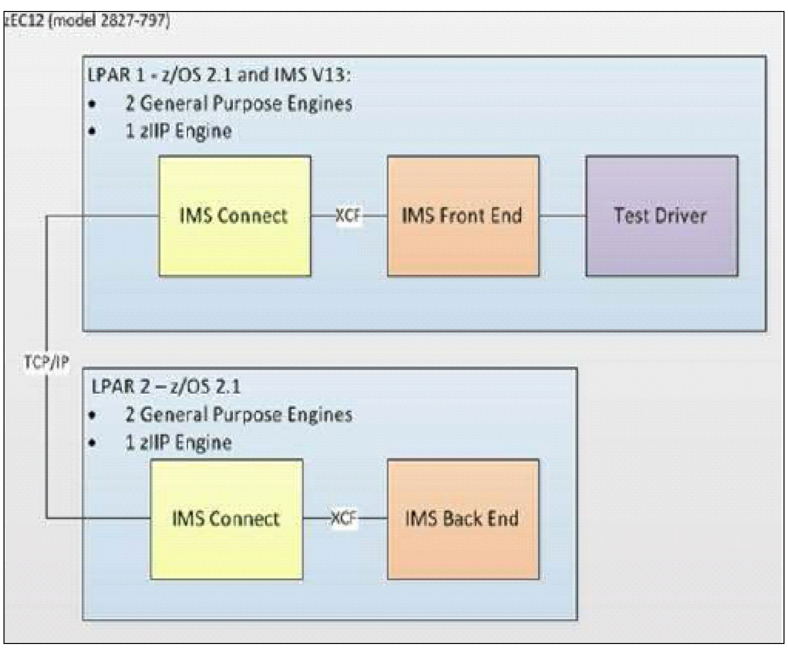


Figure 9-22 MSC zIIP utilization test configuration

The MSC Workload uses TPNS to drive 8000 terminal clients to send IMS Full Function transactions from the front end (FE) IMS on LPAR 1 to the back end (BE) IMS on LPAR 2 to be processed. Consider the following points:

- Transactions are driven through the FE IMS on LPAR 1.
- IMS Connect on LPAR 1 forwards the request to the remote IMS Connect on LPAR 2 via the MSC over TCP/IP function.
- The BE IMS on LPAR 2 processes the transaction request in 128 MPP regions.
- The transaction response follows the same path back to the terminal client.

Portions of the IMS Connect handling of MSC message threads can run on zIIP. This topology is used to measure the GP and zIIP service time when zIIP utilization support is enabled [CONDSRB(ALWAYS,HWS)] or disabled [CONDSRB(NEVER,HWS)] for the IMS Connect address space.

Evaluation results of the MSC workload zIIP utilization

In this section, we provide the MSC Workload processing measurements.

Front-end IMS system

The MSC Workload processing measurements for the front-end IMS system on LPAR 1 show the following results:

- A 4.58% savings in GP microseconds that are used per transaction when it is running with zIIP utilization enabled, compared with running without zIIP utilization enabled.
- A 0.85% increase in total CPU (GP + zIIP) microseconds that are used per transaction when it is running with zIIP utilization enabled, compared with running without zIIP utilization enabled.

Figure 9-23 shows the MSC workload result for the front-end IMS system.

zIIP Utilization Evaluation for MSC Workload for Front End IMS				
	zIIP Disabled	**zIIP Enabled**	**Delta**	**Delta %**
ETR (Tran/Sec)	2846.47	2848.43	1.96	0.07%
General CPU Busy %	49.17%	46.95%	-2.22%	-4.51%
zIIP Busy %	3.55%	8.93%	5.38%	151.54%
ITR	5789.03	6066.94	277.91	4.80%
General CPU µs/Tran	345.48	329.66	-15.83	-4.58%
zIIP CPU µs/Tran	12.47	31.35	18.88	151.38%
Total CPU µs/Tran	357.95	361.01	3.05	0.85%

Figure 9-23 MSC workload evaluation results for front-end IMS

Back-end IMS system

The MSC Workload processing measurements for the back-end IMS system on LPAR 2 show the following results:

- A 5.01% savings in GP microseconds that are used per transaction when it is running with zIIP utilization enabled, compared with running without zIIP utilization enabled.
- A 1.42% reduction6 in total CPU (GP + zIIP) microseconds that are used per transaction when it is running with zIIP utilization enabled, compared with running without zIIP utilization enabled.

Figure 9-24 shows the MSC workload result for the back-end IMS system.

zIIP Utilization Evaluation for MSC Workload for Back End IMS				
	zIIP Disabled	**zIIP Enabled**	**Delta**	**Delta %**
ETR (Tran/Sec)	2846.47	2848.43	1.96	0.07%
General CPU Busy %	74.77%	71.07%	-3.7%	-4.94%
zIIP Busy %	3.55%	8.88%	5.33%	150.14%
ITR	3806.96	4007.92	200.96	5.25%
General CPU µs/Tran	525.35	499.01	-26.34	-5.01%
zIIP CPU µs/Tran	12.47	31.18	18.70	149.97%
Total CPU µs/Tran	537.82	530.19	-7.64	-1.42%

Figure 9-24 MSC workload evaluation results for back-end IMS

9.6.5 ISC workload zIIP utilization

In this section, we describe the performance characteristics of zIIP utilization ISC workload. For more information, see 3.4, “Intersystem Communication over TCP/IP” on page 84.

Environment

Evaluations were run with IMS Connect and a back-end IMS on one LPAR, and with front-end CICS and TPNS test driver on a separate LPAR.

As shown in Figure 9-25 on page 256, the test was configured with the following components:

- Two General Purpose processors and two zIIP processors on LPAR 1 with ICON and back-end IMS.
- Two General Purpose processors with front-end CICS and test driver on LPAR 2.

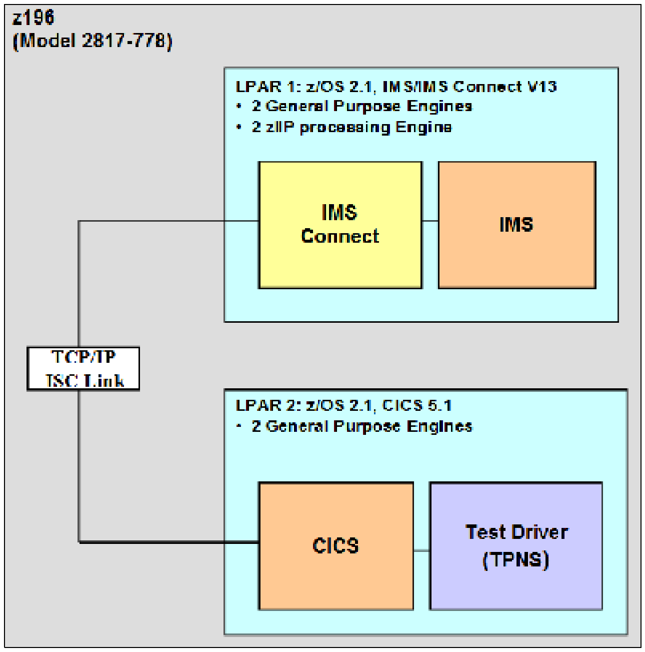


Figure 9-25 ISC zIIP utilization test configuration

TPNS was used to drive a COBOL program and to start the transactions into CICS. The CICS programs use the predefined ISC setup to then schedule IMS applications across the ISC links. IMS accepts the calls and runs the appropriate IMS transactions. The IMS applications were set up to echo the received messages back to CICS.

The ISC workload used the following procedure:

1. Clear SMF data sets.
2. Start SCI, OM, and IRLM.
3. Start IMS, MPP Regions, and IMS Connect.
4. Start CICS and verify that links are established.
5. Start TPNS workload Driver.
6. Wait a couple of minutes for workload to ramp up.
7. Switch to a clean SMF data set.
8. Start RMF I and III.
9. Wait for RMF interval report to print.
10. Issue SWI OLDS and CHECKPOINT STATISTICS.
11. Wait for next RMF interval to print.
12. Issue SWI OLDS and CHECKPOINT STATISTICS.
13. Stop RMF.
14. Switch SMF.
15. Stop TPNS drivers.
16. Stop the environment.

Evaluation results of the ISC workload zIIP utilization

The ISC workload processing measurements in the lab environment demonstrated a 4.85% savings in GP microseconds that were used per transaction when it was running with zIIP processors enabled, compared with running without the zIIP processors enabled.

Figure 9-26 on page 257 shows the ISC workload results for this evaluation.

ISC TCP/IP zIIP		
Test Case ID -->	CONDSRB=NEVER	CONDSRB=ALWAYS
Type of Channel Paths (e.g. - FICON)		
Number of Channel Paths (e.g. - 8)		
Number of CF's		
CF Level		
Workload Type	ISC TCPIP CICS->IMS	ISC TCPIP CICS->IMS
Number of concurrent IP clients	10,000	10,000
Run Data		
Run Date	12/20/2013	12/20/2013
Time Start	14:26:00	13:51:59
Time End	14:28:00	13:53:59
Duration (seconds)	119.98	120
CPU Information		
General CPU Busy %	70.95%	69.33%
zIIP CPU Busy %	5.05%	8.33%
Transaction Rates		
ITR	3230.68	3395.50
ETR	2292.17	2354.1
Programs Ended	275014.5566	282444.918
General CPU µs used each second	1419000	1386600
zIIP CPU µs used each second	101000	166600
General CPU µs/tran	619	589
zIIP CPU µs/tran	44	71
Total CPU µs/tran	**663**	**660**
Delta from None zIIP		-0.50%
Delta from None zIIP GP only		-4.85%
ITR Delta from None zIIP zIIP only		5.10%

Figure 9-26 ISC workload evaluation results

9.6.6 Summary

zIIPs can be used to grow, develop, or port certain new distributed applications on IMS in a cost-effective way.

Use of one or more zIIP specialty engines can enable cost-effective solutions for some IMS workloads. Customers can use IMS 13 and zIIP engines to potentially reduce GP usage.

Figure 9-27 shows the use of one or more zIIPs for the IMS workloads.

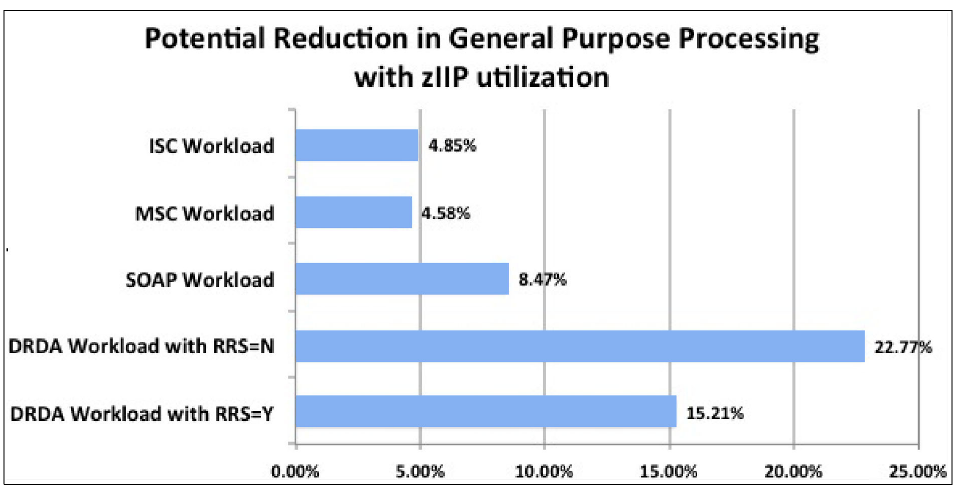


Figure 9-27 IMS 13 potential reduction in general-purpose processing with zIIP Utilization

While your results can vary, results of controlled laboratory testing by IBM demonstrated that the utilization of zIIPs might reduce GP processor processing (microseconds per transaction) of the following percentages:

- DRDA workloads by up to 15.21%
- SOAP workloads by up to 8.47%
- MSC over TCP/IP workloads by up to 4.58%
- ISC workloads by up to 4.85%

9.7 Improved SVC Directory Tables dump formatting

In addition to improving the processing efficiency in DFSCPY00 (Interregion Communication Copy) in the SVC directory entry search algorithm, IMS 13 allows the SVC directory entry blocks to be formatted by the IMS dump formatter.

The formatter is started by selecting the vector table directory (VTDIR) option on the Enhanced Dump Analysis (IMS EDA) systems panel by completing the following steps:

1. Select **E** for EDA formatting on the IMS Dump Formatting primary menu.
2. Select **5** for systems.
3. Select **VTDIR** for SVC vector table directory entries.

Figure 9-28 shows the SYSTEMS FORMATTING OPTIONS panel.

```
------------------------- SYSTEMS FORMATTING OPTIONS ------ Row 1 to 21 of 21

      S = Select                         Select choice and hit enter to process.
      T = Select (Time formatting)       Use UP/DOWN to scroll.


Cmd  Type        Description
v--------------------vvvvvvvv-----------------------------------------------
_      BCB         BCB statistics summary
_      BPECSCD     BPE LFS CSCD in the IMS control region
_      BPEHASH     BPE LFS hash tables in the IMS control region
_      CDE         CDE/SDE storage List
_      CDECOMM     CDE/SDE storage list (common only)
_      CSLA        CSL anchor block formatting
_      DFA         Definition Anchor Block and sub-blocks
_      GRMB        Global RESMGR Block, trace table, and SSCTs
_      IMODTRAC    IMODULE internal trace tables
_      LOGGER      Logger LCD blocks and buffers
_      OCMD        OM command instance block formatting
_      PDIR        PSB Directory Formatting
_      PST         PST formatting menu
_      SMB         SMB formatting menu
_      STATS       IMS statistics
_      STE         Storage Tracking Element blocks
_      TRC         Trace control blocks
_      USRX        User Exit blocks
s      VTDIR       SVC vector table directory entries
_      WRITE       Extract and write dump data to external dataset
_      XCF         IMS XCF connection blocks
****************************** Bottom of data ********************************
```

Figure 9-28 SYSTEMS FORMATTING OPTIONS panel

VTDIR Formatting MIN (M) option

To see an overview of the SVC directory tables, you can select the VTDIR option with an M instead of an S. M requests MIN formatting, which often is a smaller or a distilled set of formatting relative to S. Consider the following points:

M shows the information in a table summary format, with each VTDENTRY on its own line. This format is useful for getting a high-level picture of the entries.

Each table of 25 entries is formatted with a summary line for the table header and then a line for each entry.

Figure 9-29 shows the result panel that uses the VTDIR formatting MIN (M) option.

```
*DFSVTDIR at 00FB8B20, IMS 13.1, # Entries=25, Entry Length=30x

VTDENTRY  IMID Typ C Ver  ASID Job/STC      ASCBINTS Time           ASCBINTS          ASCB     TCB      IRCVECT  LocSrvTb IDT Trac
--------  ---- --- - ---- ---- -------- ---------------------- ---------------- -------- -------- -------- -------- --------
00FB8B50: IMS1 Ctl @ 13.1 014E IMS130YC 2013-01-10 10:42:02.97 CAC03D308CA5B477 00FBB400 006FF110 00FC1480 00000000 00000000
00FB8B80: IMS1 Dep   13.1 0150 DLIE11P2 2013-01-10 10:42:05.65 CAC03D331A2C6335 00FB8800 006FF110 00FC1480 00000000 00000000
00FB8BB0: IMS1 Dep   13.1 0151 DBRCERES 2013-01-10 10:42:05.68 CAC03D3320D7907A 00FB8680 006FF110 00FC1480 00000000 00000000
00FB8BE0: IMS1 Dep   13.1 0154 IMSBATCH 2013-01-10 10:42:32.96 CAC03D4D25938031 00FB8200 006FF208 00FC1480 006C8448 26EF20C0
00FB8C10: IMS1 Dep   13.1 0039 MPRE0200 2013-01-10 11:38:43.46 CAC049DB83074270 00F95480 006FF3A0 00FC1480 006B5448 26EF2180
00FB8C40: IMS1 Dep   13.1 003A MPRE0100 2013-01-10 11:38:43.52 CAC049DB90DB6BB0 00F95300 006FF3A0 00FC1480 006B5448 26EF2240
00FB8C70: IMS1 Dep   13.1 0020 BMP1ASCB 2013-01-10 11:38:45.60 CAC049DD8C260BF8 00FC1700 006CF998 00FC1480 006B1448 26EF2300
00FB8CA0: IMS1 Dep   13.1 0065 BMP2ASCB 2013-01-10 11:38:45.67 CAC049DD9EF20974 00FC1080 006CF998 00FC1480 006B1448 26EF23C0
00FB8CD0: IMS1 Dep   13.1 0067 BMP3ASCB 2013-01-10 11:38:45.75 CAC049DDB0190934 00FC0A00 006CF998 00FC1480 006B1448 26EF2480
00FB8D00: IMS1 Dep   13.1 002D BMP4ASCB 2013-01-10 11:38:45.79 CAC049DDBAF5CFB4 00FC0580 006CF998 00FC1480 006B1448 26EF2540
00FB8D30: IMS1 Dep   13.1 003C MPRE0151 2013-01-10 11:38:43.70 CAC049DBBC1191B8 00F95000 006FF3A0 00FC1480 006B5448 26EF2600
00FB8D60: IMS1 Dep   13.1 003B MPRE0251 2013-01-10 11:38:43.58 CAC049DB9F3812F0 00F95180 006FF3A0 00FC1480 006B5448 26EF26C0
00FB8D90: IMS1 Dep   13.1 003D MPRE0202 2013-01-10 11:38:43.74 CAC049DBC707C838 00FA2E80 006FF3A0 00FC1480 006B5448 26EF2780
00FB8DC0: IMS1 Dep   13.1 003E MPRE0M1A 2013-01-10 11:38:43.79 CAC049DBD1F0FA34 00FA2D00 006FF3A0 00FC1480 006B5448 26EF2840
00FB8DF0: IMS1 Dep   13.1 003F MPRE0102 2013-01-10 11:38:43.83 CAC049DBDD07B774 00FA2B80 006FF3A0 00FC1480 006B5448 26EF2900
00FB8E20: IMS1 Dep   13.1 0040 MPRE0253 2013-01-10 11:38:43.88 CAC049DBE7F20F70 00FA2A00 006FF3A0 00FC1480 006B5448 26EF29C0
00FB8E50: IMS1 Dep   13.1 0042 MPRE0153 2013-01-10 11:38:43.97 CAC049DBFDB58B75 00FA2700 006FF3A0 00FC1480 006B5448 26EF2A80
00FB8E80: IMS1 Dep   13.1 0041 MPRE0M2A 2013-01-10 11:38:43.92 CAC049DBF2C5B638 00FA2880 006FF3A0 00FC1480 006B5448 26EF2B40
00FB8EB0: IMS1 Dep   13.1 0043 MPRE0204 2013-01-10 11:38:44.01 CAC049DC07F90E78 00FA2580 006FF3A0 00FC1480 006B5448 26EF2C00
00FB8EE0: IMS1 Dep   13.1 0044 MPRE1M1A 2013-01-10 11:38:44.05 CAC049DC127D19F8 00FA2300 006FF3A0 00FC1480 006B5448 26EF2CC0
00FB8F10: IMS1 Dep   13.1 009D MPRE0104 2013-01-10 11:38:44.10 CAC049DC1DA34274 00FA2000 006CF998 00FC1480 006B5448 26EF2D80
00FB8F40: IMS1 Dep   13.1 009C MPRE0255 2013-01-10 11:38:44.14 CAC049DC2941A230 00F53000 006CF998 00FC1480 006B5448 26EF2E40
00FB8F70: IMS1 Dep   13.1 009E MPRE1M2A 2013-01-10 11:38:44.19 CAC049DC358C0278 00FA5B00 006CF998 00FC1480 006B5448 26EF2F00
00FB8FA0: IMS1 Dep   13.1 0097 MPRE0155 2013-01-10 11:38:44.23 CAC049DC3F358674 00FA5680 006CF998 00FC1480 006B5448 26EF2FC0
00FB8FD0: IMS1 Dep   13.1 0099 MPRE0206 2013-01-10 11:38:44.27 CAC049DC48E68C30 00FA5500 006CF998 00FC1480 006B5448 26DAC0C0
```

Figure 9-29 Result panel that uses the VTDIR formatting MIN (M) option

VTDIR formatting normal (S) option

If you need more information, you can select the option VTDIR with an S and you see each VTDPREFIX and VTDENTRY entry in a field-by-field format.

Figure 9-30 shows the result panel that uses the VTDIR formatting normal (S) option.

```
*******************************************************************
* DFSVTDIR at 00FB8B20, IMS 13.1, # Entries=25, Entry Length=30x *
*******************************************************************
 VTDPREFX: 00FB8B20
    +0000  ID....... DFSVTDIR  REL...... 1310      RESERVED. 0000      NBLK..... 00FB3B20  ENT...... 00FB8B50  ECNT..... 0019
    +0016  ELEN..... 0030      RESERVED. 00000000  RESERVED. 00000000  RESERVED. 00000000  RESERVED. 00000000  RESERVED. 00000000
    +002C  RESERVED. 00000000

*----------------------------------------------------------*
* VTDE at 00FB8B50 for IMS 13.1 IMS1 Type=Ctl ASID=014E   *
* VTDEINTS = 2013-01-10 10:42:02.975899 Job/STC=IMS130YC *
*----------------------------------------------------------*
 VTDENTRY: 00FB8B50
    +0000  NEXT..... 00FB8B80  IMID..... IMS1      ASCB..... 00FBB400  RIND..... 01        CRC...... 7C        REL...... 1310
    +0010  SVTA..... 00FC1480  LOCL..... 00000000  INTS..... CAC03D30  8CA5B477            TCB...... 006FF110  ASID..... 014E
    +0026  RESERVED. 0000      IDTT..... 00000000  RESERVED. 00000000
    Flag analysis for vtdexind:
      vtdexctl (01x) - Control region

*----------------------------------------------------------*
* VTDE at 00FB8B80 for IMS 13.1 IMS1 Type=Dep ASID=0150   *
* VTDEINTS = 2013-01-10 10:42:05.652742 Job/STC=DLIE11P2 *
*----------------------------------------------------------*
 VTDENTRY: 00FB8B80
    +0000  NEXT..... 00FB8BB0  IMID..... IMS1      ASCB..... 00FB8800  RIND..... 04        CRC...... 00        REL...... 1310
    +0010  SVTA..... 00FC1480  LOCL..... 00000000  INTS..... CAC03D33  1A2C6335            TCB...... 006FF110  ASID..... 0150
    +0026  RESERVED. 0000      IDTT..... 00000000  RESERVED. 00000000
    Flag analysis for vtdexind:
      vtdexdep (04x) - Dependent region
```

Figure 9-30 Result panel that uses the VTDIR formatting normal (S) option

9.8 Performance enhancements conclusion

IMS 13 delivers excellent performance, scalability, availability, and security with reduced resource usage compared to IMS 12.

Performance evaluations show the significant cost and contention reduction features that were implemented for IMS Universal Database Driver, IMS Connect/OTMA, Log Latch Rewrite, Block Serialization, and zIIP utilization.

The enhancements provide improved efficiency in core IMS and z/OS processing as well. IMS customers can see benefit from the performance enhancements that are featured in IMS 13 for all areas, including Transaction Manager, Database Manager, and Systems components.

As evident from the results, IMS 13 demonstrates an increase in CPU efficiency per transaction in nearly all function areas that are measured, including Fast Path, Full Function, Shared Queues, and CICS - IMS DBCTL.

Figure 9-31 shows a summary of the ITR performance improvements in IMS 13 from IMS 12. The performance improvements in the base functionality came from many enhancements that were implemented in the new release, such as improved IMS application call copy processing (DFSCPY00), IMS Connect/OTMA improvements, log latch contention reduction, and optimized Fast Path syncpoint processing.

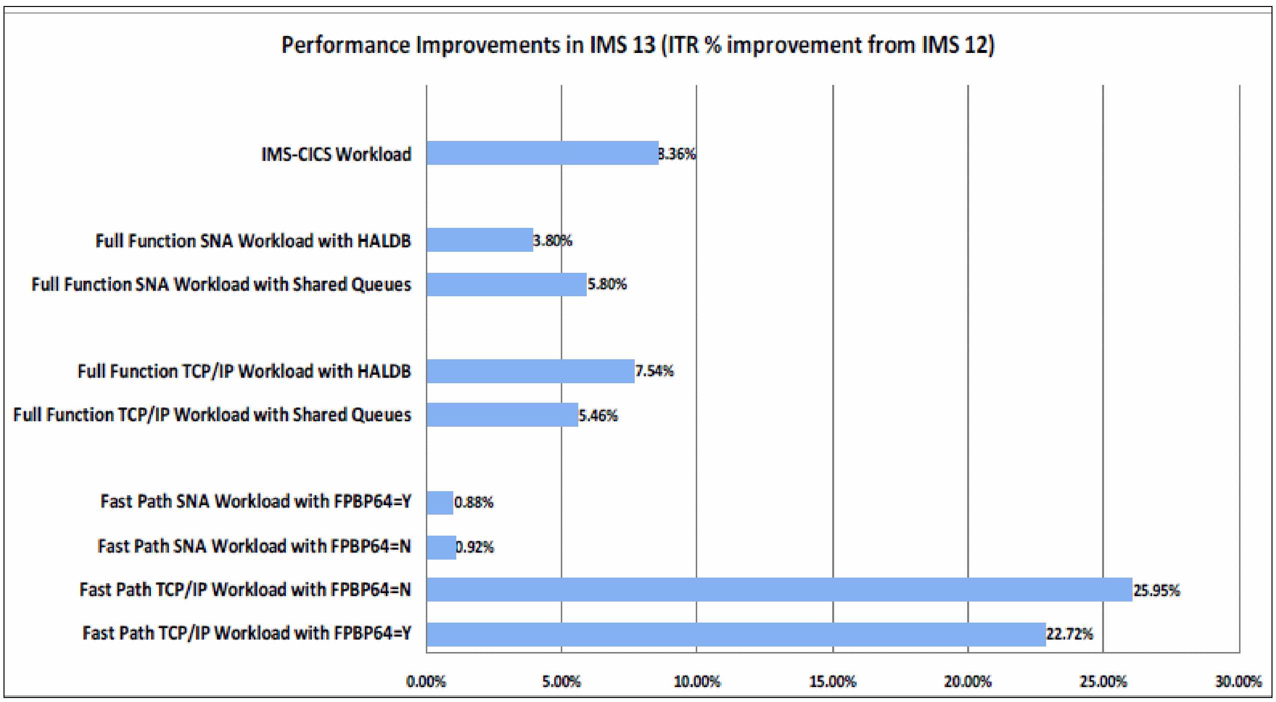


Figure 9-31 Performance improvements in IMS 13 from IMS 12

Additionally, many small performance improvement-related changes were made throughout the IMS system code. Although individually these changes were small, they cumulatively contributed to the overall improvement in IMS 13 processing efficiency.

10

Installation and migration considerations

This chapter describes the installation and migration tasks to consider if you are migrating from a supported release to IMS 13.

This chapter includes the following topics:

- Packaging, prerequisites, and coexistence
- IMS library updates
- Installation verification program
- Syntax Checker
- Installation and migration tasks
- Migration considerations
- Review of migration considerations

10.1 Packaging, prerequisites, and coexistence

This section describes the packaging, prerequisites, and coexistence for IMS 13. For more information, see the following publications:

- *Program Directory for Information Management System Transaction and Database Servers V13.0*, GI10-8914
- *IMS Version 13 Release Planning*, GC19-3658

10.1.1 Packaging

The IMS product is packaged under several function modification identifiers (FMIDs). This packaging choice is in response to IMS internal requirements and is subject to change in the future.

The following IMS 13 packaging is the same as that for IMS 11 and IMS 12:

- Transaction Manager is a prerequisite for ETO.
- Recovery Level Tracking RSR is a prerequisite for Database Level Tracking RSR.
- IRLM 2.3 (FMID HIR2230) is the only IRLM shipped and supported by IMS 13 and works with all supported releases of IMS.

The IMS 13 program number is 5635-A04, Component ID 5635A0400. The IMS 13 Component ID for RETAIN® is 5635A0400.

All FMIDs that are installed conform to the following packaging standards:

- Via CBPDO, ServerPac, or CustomPac
- The use of SMP/E RECEIVE, APPLY, ACCEPT
- Outside of the Installation Verification Program (IVP)

10.1.2 Distribution media

The IMS product can be distributed by using the following methods:

- Custom-Built Product Delivery Offering (CBPDO)

  CBPDO is a package that includes IMS and can include other products in the same System Release (SREL).

  This media contains all products and features (FMIDs) combined on one logical tape. Depending on the size, it might be multiple physical tapes. It is a DBS feature distribution tapes.

- ServerPac

  ServerPac is an entitled software delivery package. It consists of products and service for which IBM performed the SMP/E installation steps and some of the post-SMP/E installation steps. It is a DBS feature pre-built SMP/E, DLIBs, and TLIBs.

- RSU

  Service integrated monthly. Beyond this monthly base, highly pervasive (HIPER) and programming error (PE) resolution PTFs are integrated. Other services are provided on service tape.

  Target and Distribution libraries are shipped. Complete service integration is provided twice yearly in March and September.

- SystemPac

  DBS Feature can be customized. Custom systems are built and delivered to customers. Service is integrated to the recommended service level. Target and Distribution libraries are shipped.

- SUP

  SUP is a roll-up of maintenance into the FMIDs with ease installation. Customers do not need to reinstall and the maintenance continues to be distributed as usual.

  It can be used as a way to upgrade maintenance. All product orders that are filled after the SUP is available are built at the SUP level. PSP buckets contain the details of the SUP and the list of APARs that are included in the FMIDs.

Delivery schedule

ShopZseries delivery schedules can be found at this website:

https://www14.software.ibm.com/webapp/ShopzSeries/ShopzSeries.jsp?action=glcrad

10.1.3 Preventive Service Planning

Before you begin installing IMS 13, review the current Preventive Service Planning (PSP) information.

The PSP buckets maintain lists (which are identified since the package was created) of any recommended or required service for the installation of this package. This service includes software PSP information that contains HIPER and required program temporary fixes (PTFs) against the base release.

If you obtained IMS 13 as part of a CBPDO, HOLDDATA is included.

If the CBPDO for IMS 13 is older than two weeks by the time you install the product materials, contact the IBM Support Center or use S/390® SoftwareXcel to obtain the latest PSP bucket information.

You can also obtain the latest PSP bucket information by seeing the Preventive Service Planning bucket page at this website:

http://www14.software.ibm.com/webapp/set2/psearch/search?domain=psp

For program support, see this Software Support website:

http://www.ibm.com/software/support/

Upgrades and subsets

PSP buckets are identified by Upgrades, which specify product levels, and Subsets, which specify the FMIDs for a product level.

Table 10-1 on page 264 shows the Upgrade and Subset values for IMS 13.

Table 10-1 IMS 13 PSP Upgrade and Subset ID

Upgrade	Subset	Description
IMS1300	HMK1300/GA	IMS V13.01.00 System Services
IMS1300	JMK1301/GA	IMS V13.01.00 Database Manager
IMS1300	JMK1302/GA	IMS V13.01.00 Transaction Manager
IMS1300	JMK1303/GA	IMS V13.01.00 Extended Terminal Option
IMS1300	JMK1304/GA	IMS V13.01.00 Recovery Level Tracking
IMS1300	JMK1305/GA	IMS V13.01.00 DB Level Tracking
IMS1300	JMK1306/GA	IMS V13.01.00 Java on-demand feature
IMS1300	HIR2230/1239	Internal Resource Lock Manager (IRLM) V2.3

10.1.4 Support procedures

Table 10-2 identifies the function modification IDs (FMID) and component IDs (COMPID) for IMS 13 for use when you must report any problems to your IBM Support Center.

Table 10-2 IMS 13 component IDs

FMID	COMPID	Component Name	RETAIN Release
HMK1300	5635A0400	System Services	300
JMK1301	5635A0400	Database Manager	301
JMK1302	5635A0400	Transaction Manager	302
JMK1303	5635A0400	Extended Terminal Option	303
JMK1304	5635A0400	RSR Recovery-Level Tracking	304
JMK1305	5635A0400	RSR Database-Level Tracking	305
JMK1306	5635A0400	IMS Java on-demand	306
HIR2230	569516401	Internal Resource Lock Manager V02.03.00	230

10.1.5 SMP/E processing

Always RECEIVE current Enhanced HOLDDATA before SMP/E processing.

For more information about Enhanced Holddata, see this website:

http://service.boulder.ibm.com/390holddata.html

Be sure to resolve PEs during processing. Contact the IBM Support Center for assistance as needed.

SMP/E and CBPDO setup jobs

Program Directory provides a JCL to unload sample jobs to perform SMP/E processing and set up. Each job contains instructions for customization.

Customization of the sample jobs

Example 10-1 shows there are same variables in multiple jobs so you might want to save in a separate file and copy information.

Example 10-1 How to change same variables in multiple jobs

```
C   #globalcsi IMS.V13.GLOBAL.CSI   all
C   targlib    IMS13T               all
```

Figure 10-1 shows an example of the instructions for customization of the sample jobs.

```
//* NOTES:                                                           */
//* 1) REVIEW THE SMP CONTROL STATEMENTS BEFORE SUBMITTING           */
//*    THIS JOB.                                                     */
//* 2) ADD A JOB CARD TO MEET YOUR SYSTEM'S REQUIREMENTS.            */
//* 3) CHANGE #GLOBALCSI TO THE DATASET NAME OF YOUR GLOBAL          */
//*    CSI DATA SET.                                                 */
//* 4) CHANGE TARGLIB TO THE NAME OF YOUR TARGET ZONE.               */
//* 5) REMOVE THOSE FMIDS THAT YOU DO NOT WISH TO APPLY FROM         */
//*    THE SELECT AND FORFMID OPERANDS OF THE APPLY STATEMENT.       */
//*           HMK1300  <======= FMID TO BE INSTALLED                 */
//*           HIR2230  <======= FMID TO BE INSTALLED                 */
//*           JMK1301  <======= FMID TO BE INSTALLED                 */
//*           JMK1302  <======= FMID TO BE INSTALLED                 */
//*           JMK1303  <======= FMID TO BE INSTALLED                 */
//*           JMK1304  <======= FMID TO BE INSTALLED                 */
//*           JMK1305  <======= FMID TO BE INSTALLED                 */
//*           JMK1306  <======= FMID TO BE INSTALLED                 */
...
...
```

Figure 10-1 Example instructions for customization of the sample jobs

Important: Be sure to specify CAPS ON as appropriate. HFS path names must be in lowercase characters.

SMP/E and CBPDO jobs

Table 10-3 lists the SMP/E and CBPDO setup and sample jobs that are shipped with IMS 13.

Table 10-3 SMP/E and CBPDO setup and sample jobs that are shipped with IMS 13

Job name	Job type	Description
DFSALA	SMP/E	Allocate and start a new SMP/E CSI data set (optional but recommended).
DFSALB	SMP/E	Initialize SMP/E zones that are allocated (optional but recommended). Ensure that ACCJCLIN is specified in DLIB zone before processing FMIDs.
RECEIVE	SMP/E	Receive job that is provided by CBPDO. Program Directory contains instructions for obtaining the JCL.
DFSALLOC	ALLOCATE	Allocate target and distribution libraries.
DFSZFSAL	ALLOMZFS	Allocate a ZFS file system (optional).

Job name	Job type	Description
DFSJSMKD	MKDIR	Start DFSJMKDR EXEC to allocate ZFS/HFS paths for IMS Java.
DFSDDEF1	DDDEF	Define SMP/E DDDEFs for core IMS.
DFSDDEF2	DDDEF	Define SMP/E DDDEFs for IMS on-demand.
DFSAPPLY	SMP/E	SMP/E APPLY of FMIDs and service. Be sure to resolve PEs.
DFSACCEP	SMP/E	SMP/E ACCEPT of FMIDs and service. Be sure to resolve PEs.

Optional sample CBPDO jobs

The following optional jobs are provided to install IMS in its own unique SMP/E environment (GLOBAL Zone):

- DFSALA

  Allocate and initialize new CSI.

- DFSALB

  Initialize CSI zones, allocate SMP/E data sets, and build DDDEF entries for SMP/E.

If these jobs are *not* used, be sure that ACCJCLIN is set in the IMS distribution zone before ACCEPT processing.

Also, ensure that ACCJCLIN is set in sample job DFSALB and SMP/E OPTION and UTILITY entries are added in sample job DFSALB.

File system paths changes

The file system paths to install IMS 13 changed to be more efficient during the migration process. Table 10-4 shows how the file system paths differ between IMS 12 and IMS 13.

Table 10-4 IMS 13 and IMS 12 file system paths changes

DDNAME	IMS 13 path name	IMS 12 path name
SDFSIC4J	/usr/lpp/ims/ims13/ico/IBM/	/usr/lpp/ims/ims12/ico/IBM/
SDFSJHFS	/usr/lpp/ims/ims13/imsjava/IBM/	/usr/lpp/ims/ims12/imsjava/IBM/
SDFSJRAR	usr/lpp/ims/ims13/imsjava/rar/IBM/	/usr/lpp/ims/ims12/imsjava/IBM/
SDFSJ2CI	/usr/lpp/ims/ims13/imsjava/cics/IBM/	/usr/lpp/ims/ims12/imsjava/cics/IBM/
SDFSJCIC	/usr/lpp/ims/ims13/imsjava/classic/cics/IBM/	/usr/lpp/ims/ims12/imsjava/classic/cics/IBM/
SDFSJCPI	/usr/lpp/ims/ims13/imsjava/classic/IBM/	/usr/lpp/ims/ims12/imsjava/classic/IBM/
SDFSJSAM	/usr/lpp/ims/ims13/imsjava/ivp/IBM/	/usr/lpp/ims/ims12/imsjava/ivp/IBM/
SDFSJCPS	/usr/lpp/ims/ims13/imsjava/classic/ivp/IBM/	/usr/lpp/ims/ims12/imsjava/classic/ivp/IBM/

Note: More system paths can be added by maintenance.

For more information about cleaning up obsolete data sets, paths, and DDDEFs that were created and used by previous releases of IMS and are no longer used in IMS 13, see *Program Directory for Information Management System Transaction and Database Servers V13.0*, GI10-8914.

You can delete these objects after you delete the previous release from your system.

10.1.6 Software prerequisites

IMS 13 has base software requirements. Some individual functions have more software requirements.

Operating software requirements

IMS 13 and its various functions have specific operating software requirements.

Before you install IMS 13, check with your IBM Support Center or the Information or Access or Service Link for more preventive service planning (PSP) information of which you must be aware. The PSP upgrade name for IMS Version 13 is IMS1300.

For more information about the z/OS service levels that are required for installation and execution, see *Program Directory for Information Management System Transaction and Database Servers V13.0*, GI10-8914.

Minimum software level

IMS Version 13 requires the following minimum version, release, or modification levels:

- z/OS V1R13 (5694-A01)

  z/OS 1.13 is the base operating system level for IMS 13 and includes the following components:

  - RACF (included in separately orderable SecureWay Security Server), or equivalent, if security is used
  - High Level Assembler Toolkit Release 5 (5696-234)
  - APARs/PTFs OA39392/UA66823, OA36172/UA61786
  - If zIIP offload is used: OA39392/UA66823

- IRLM 2.3

  If the IRLM is used, IRLM Version 2.3 is delivered with IMS 13.

10.1.7 Minimum software levels for optional functions

In this section, we describe the software levels that are required by optional IMS functions.

Java Dependent Regions

The Java Dependent Regions enhancement requires JDK 6.0.1.

ISC TCP/IP requirements

IMS 13 ISC TCP/IP function includes the following software requirements:

- IBM CICS Transaction Server for z/OS, Version 5.1 (or later).
- IMS Connect to provide TCP/IP socket connection support for IMS.
- The Common Service Layer (CSL) with at least the Structured Call Interface (SCI) and the Operations Manager (OM). SCI is required for communications between IMS and IMS Connect and OM is required for type-2 command support.

- A single point of control (SPOC) program, such as the IMS TSO SPOC, to issue type-2 commands to the OM API or REXX SPOC API.
- For each IMS subsystem that uses dynamically defined terminals with ISC TCP/IP, the IMS Extended Terminal Option (ETO) is required.

Native SQL for COBOL

Native SQL for COBOL requires COBOL 5.1 and PM92523/UK98481, which adds the IMS co-processor function.

IMS 13 requires APAR PM97137 (PTF UK98028) for preconditioning code and APAR UK98418 (PTF UK98418) for logger abend.

IMS Universal drivers

Depending on the environment, the IMS Universal drivers require the following software products level:

- IBM JDK 6.0.1 or later
- DB2 V9.1 or later
- WebSphere V7 or later
- CICS V4.1 or later

IMS Enterprise Suite

.NET Data Provider in IMS Enterprise Suite 2.3 requires an implementation of CSL, IMS Connect, ODBM, and Catalog.

Database versioning

The IMS 13 database versioning enhancement requires the implementation of the IMS Catalog.

IMS 13 interactions

IMS 13 supports interactions with the following software products level:

- DB2: Versions 9, 10, and 11
- CICS: Versions 3.2, 4.1, and 4.2, 5.1

  CICS 5.1 is required for ISC TCP/IP support and requires z/OS 1.13.
- WebSphere MQ for z/OS: Versions V7.0.1, V7.1:
  - WebSphere MQ V6 end of service was September 2012
  - 7.01 introduced WebSphere MQ message expiry interfacing with IMS transaction expiration
  - 7.1 enhanced the expiry support and support for OTMA resource monitoring protocol messages

10.1.8 Hardware prerequisites

In this section, we describe the IMS 13 base hardware requirements. (Some individual functions have more hardware requirements.)

Processor requirements

IMS 13 runs only on 64-bit processors that are running in z/Architecture® mode, which is not new for IMS. Consider the following points:

- Processors must also support the Long-Displacement Facility of the z/Architecture
- ESA mode is not supported

For a list of zSeries machines, see this website:

http://www.ibm.com/systems/z/hardware/

Note: z900 machines must be at GA2 level (microcode level 3G or later).

Coupling facility requirements

IMS 13 has the following coupling facility requirements:

- Sysplex Data Sharing (including Data Caching and VSO Data Sharing)
- Coupling Facility (CF) level 9, or later

Shared queues and shared EMH support

For shared queues and EMH, Coupling Facility level 9 or later is needed.

For system-managed CF duplexing, CF level 12 is needed, or later and bidirectional CF to CF links.

EAV support for non-VSAM data sets

EAVs are supported on DS8000® at microcode level R4.0 via bundle 64.0.175.0 or higher.

10.1.9 Prerequisite maintenance

You can identify the outstanding service that must be installed on your IMS 11 and IMS 12 systems to enable them to coexist with IMS 13 by using SMP/E and the IMS 13 FIXCAT category.

The FIXCAT category for IMS Version 13 is IBM.Coexistence.IMS.V13 with the keyword `IMSV13COEX`.

The following example shows the steps that are used to determine what IMS service must be installed on IMS 11 or IMS 12 to coexist with IMS 13:

1. Download the current enhanced HOLDDATA.
2. Issue SMP/E RECEIVE for the current enhanced HOLDDATA.
3. Run the **`SMP/E REPORT MISSINGFIX`** command that is pointing to your IMS 11 or IMS 12 zone.

   Example 10-2 shows the SMP/E statements to run the SMP/E REPORT MISSINGFIX command.

   Example 10-2 SMP/E statements to run the SMP/E REPORT MISSINGFIX command

   ```
   SET BOUNDARY (GLOBAL).
      REPORT MISSINGFIX ZONES (targetzone)
      FIXCAT(IBM.Coexistence.IMS.V13).
   ```

FIXCAT categories

Table 10-5 lists the categories that are defined for IMS.

Table 10-5 FIXCAT categories that are defined for IMS

Category	Keyword	Description
IBM.Coexistence.IMS.V11	IMSV11COEX/K	Fixes with which IMS V9 and V10 coexist and fallback from IMS V11.
IBM.Coexistence.IMS.V12	IMSV12COEX/K	Fixes with which IMS V10 and V11 coexist and fallback from IMS V12.
IBM.Coexistence.IMS.V13	IMSV13COEX/K	Fixes with which IMS V11 and V12 coexist and fallback from IMS V13.
IBM.TargetSystem-Required Service.IMS.V11	IMSV11TGT/K	Fixes that are required on other IBM products with which they run IMS V11.
IBM.TargetSystem-Required Service.IMS.V12	IMSV12TGT/K	Fixes that are required on other IBM products with which they run IMS V12.
IBM.TargetSystem-Required Service.IMS.V13	IMSV13TGT/K	Fixes that are required on other IBM products with which they run IMS V13.

Related information

For more information, see the following resources:

- Enhanced HOLDDATA for z/OS:

  http://service.software.ibm.com/holdata/390holddata.html

- IBM Fix Category Values and Descriptions:

  http://www-03.ibm.com/systems/z/os/zos/features/smpe/fix-category.html

10.1.10 Support status of IMS versions

Table 10-6 lists the support status of latest IMS versions.

Table 10-6 Support status of latest IMS versions

Version	Generally available	End of marketing	End of support
IMS 10	October 26, 2007	September 12, 2011	November 12, 2012
IMS 11	October 30, 2009	September 9, 2013	October 30, 2014
IMS 12	October 28, 2011	Not announced	Not announced
IMS 13	October 25, 2013	Not announced	Not announced

10.1.11 IMS 13 supported connections

ISC is supported by the following releases of IMS and CICS:

- IMS 13, IMS 12, and IMS 11
- CICS Transaction Server V 4.2, V4.1, V3.2 and V3.1
- CICS Transaction Server V 5.1 for ISC TCP/IP connectivity
- User-written software

MSC connectivity is supported by all releases of IMS: IMS 13, IMS 12, and IMS 11.

Shared queues are supported by releases IMS 13, IMS 12, and IMS11 of IMS Transaction Manager.

10.1.12 Supported migrations and coexistence

Migration from previous releases IMS 11 or IMS 12 is supported.

DBRC

IMS 13 RECONs can be used by all supported IMS releases with compatibility service. As always, the following DBRC migration or coexistence SPEs are provided:

- IMS 11: APAR PM53134/UK80026
- IMS 12: APAR PM53139/UK80027

Security

In IMS 13, the SECURITY macro was removed. SPEs are provided to support coexistence and migration.

IMS 13 eliminates RCLASS and SECCNT support in Security macro and adds support as execution parameters.

The following SPEs support the removal of the SECURITY macro and the addition of the execution parameter support in previous releases:

- IMS 11: PM48203/UK74050 and PM72199/UK82616
- IMS 12: PM48204/UK74051 and PM73558/UK82617

10.1.13 IMS APARs coexistence requirements

IMS 11 and IMS 12 must have certain APARs installed to coexist with IMS Version 13. Table 10-7 lists the coexistence APARs and PTFs that are needed for various IMS functions.

Table 10-7 Coexistence APARs and PTFs

IMS 13 function	IMS 11 coexistence APAR/PTF	IMS 12 coexistence APAR/PTF
Database Change Accumulation utility (DFSUCUM0)	PM37033/UK67281	None
DBRC RECON data sets	PM53134/UK80026	PM53139/UK80027
HALDB online reorganization (OLR)	PM31791/UK67485	None
PSB library	PM23843/UK64013 PM32390/UK65495 PM35639/UK66632	None
Security enhancements	PM48203/UK74050 PM72199/UK82616	PM48204/UK74051 PM73558/UK82617

For more information about coexistence maintenance, see *IMS 13 Release Planning*, GC19-3658.

10.1.14 IMS DB log records

In this section, new or enhanced record types are listed.

New type x'2931' IMS log record

This record is written before the HALDB OLR start time so that products that are performing a point-in-time recovery (PITR) can correctly build the output (target) data sets when a PITR is done.

This record is a copy of the x'2930' log record and contains the same information, except for the Update Set ID (USID) and the Update Sequence Number (USN), which are excluded.

> **Important:** User modifications or non IBM vendor software that reference the x'2930' log record might need to be modified to also reference the x'2931' log record.

Type x'47' log record restructure

This enhancement changes the structure of type 47 log records with which records are spanned. Database names can span multiple records. This change increases scalability to with which an unlimited number of database and HALDB partitions with uncommitted updates can be logged during System Checkpoint.

10.1.15 Coexistence IMS database utilities

IMS databases can be shared between supported IMS releases.

Batch Backout, Log Archive, Log Recovery, and Log Analysis

The Batch Backout, Log Archive, Log Recovery, and Log Analysis utilities access one log.

The release level of the utility must match the IMS release that was used to create the log.

IMS 13 Database Recovery

The IMS 13 Database Recovery utility (DFSURDB0) accepts Image Copies, HISAM Unload data sets, Change Accumulation data sets, and IMS logs as inputs.

These inputs can be created by any currently supported release of IMS (IMS 13, IMS 12, and IMS 11).

IMS 13 Change Accumulation

The Change Accumulation utility accepts IMS logs and Change Accumulation data sets as inputs.

These inputs can be created by any currently supported release of IMS (IMS 13, IMS 12, and IMS 11).

10.1.16 Dynamic Resource Definition coexistence

When you are migrating DRD, the considerations that are described in this section are important.

Mixed-release Dynamic Resource Definition (DRD) environments

For an IMS 12 or IMS 13 system that uses the IMSRSC repository to coexist with an IMS 10 or 11 system that uses the RDDS data sets, the following coexistence maintenance must be installed:

- APAR/PTF PM19025 (UK63960) (IMS 10)
- APAR/PTF PM19026 (UK63964) (IMS 11)

Repository change-list enhancement

APAR PM77568 adds the repository change-list enhancement in IMS 13.

APAR PM80588 is a co-existence APAR for IMS 12 with IMS 13 and a prerequisite to APAR PM77568, which has the following description:

> “When the QUERY, UPDATE and DELETE of an IMS change list is being processed by a V12 Resource Manager (RM) address space, APAR PM80588 adds logic to disable support of unsupported functions in IMS 12.”
>
> It is required by all IMS 12 users of the IMSRSC repository in a mixed environment of IMS V12 and IMS V13
>
> If there is no IMS 12 SCI, RM, or OM in IMSplex, PM80588 is not required.

10.1.17 CSL coexistence

When a system is migrated by using the Common Service Layer (CSL) address spaces, any address space can be migrated to IMS 13 before the other address spaces are migrated.

It is permissible to have some SCI, RM, OM, ODBM, and IMS subsystems on IMS 13 while others are on IMS 12 and IMS 11.

The use of IMS 13 CSL address spaces is recommended when any IMS subsystem is at IMS 13.

Mixed-release CSL environments

If you are running a mixed version in an IMSplex, it is recommended that SCI and OM from the latest version of IMS are used

For an IMS 12 or IMS 13 system that uses the CSL RM address space to coexist with an IMS 10 or IMS 11 CSL RM address space, the following coexistence maintenance must be installed:

- PM32951 (UK68883) (IMS 10)
- PM32766 (UK68882) (IMS 11)

10.1.18 Shared Queues coexistence

IMS 13 CQS (shared queues) coexistence includes the following restrictions:

- IMS 13 cannot register with a pre-IMS 13 CQS
- IMS 12 cannot register with a pre-IMS 12 CQS
- IMS 11 or IMS 12 can register with an IMS 13 CQS

User or vendor-written CQS clients that want to register with IMS 13 CQS must use CQS macros from IMS 11, IMS 12, or IMS 13.

User or vendor-written CQS clients that are assembled with IMS 13 CQS macros cannot register with a PRE IMS 13 CQS.

An IMS 11, IMS 12, or IMS 13 CQS can connect to the same coupling facility.

In a shared queue, mixed-release environment, IMS 12 APAR PM75791 (PTF UK83421) resolves issues with downward compatibility with IMS 11.

With the fix applied, IMS 12 obtains only an extended AOS (APPC OTMA synchronous) header if all systems are IMS 12 (DBRC minvers=12.1). Otherwise, a shorter header that is compatible with IMS 11 is used.

This fix affects processing for conversational transactions or output that is greater than 61 K in mixed (IMS 11 and IMS 12) shared queues environments with an AOS specification other than “N”, and was added to the IMS 13 base.

10.1.19 ISC TCP/P restrictions

The following functions are not supported by ISC TCP/IP enhancement:

- CICS transactions that use the SEND(INVITE)/RECV protocol for synchronous communication
- CICS transactions that use the SEND(LAST)/RECV protocol
- Extended Recovery Facility (XRF)
- IMS operator commands except for /DISPLAY and /RDISPLAY
- IMS conversational mode transaction
- IMS response mode transactions, including Fast Path
- IMSplex Terminal Management (STM)
- Front-End Switch (FES)
- Message Format Service (MFS)
- VTAM Generic Resources (VGR)

10.1.20 RSR coexistence

The migration of systems that use Remote Site Recovery (RSR) is similar to migrations for previous releases.

IMS 13 tracking systems can process logs that are produced by previous releases.

The IMS 13 Isolated Log Sender (ILS) function of the Transport Manager System (TMS) can process logs that are created by previous releases.

However, IMS 12 and IMS 11 tracking systems cannot accept logs that are produced by IMS 13. Also, the IMS 12 and IMS 11 ILSs cannot accept logs that are produced by IMS 13.

10.1.21 IMS 13 log records changes

If you have application programs that process IMS log records, you should verify whether they are affected by the changes to the log records.

You can assemble DSECTs for IMS log records by using the following ILOGREC macro:

```
ILOGREC RECID=ALL
```

Table 10-8 lists the hexadecimal log record types that are new or changed in IMS 13.

Table 10-8 New or changed log records for IMS 13

Type	Description
x'0408'	Relays information to the RSR tracking site for Batch backout update UOR.
x'07'	An application program ended.
x'16'	A `/SIGN` command successfully completed.
x'22'	A type-2 command that is completed successfully and is recoverable across a restart.
x'35'	A message was enqueued or re-enqueued.
x'36'	A message was dequeued or saved or deleted.
x'4021'	A checkpoint was taken. VTAM VTCBs follow.
x'4507'	Checkpoint statistics were gathered, including statistics from the 64-bit storage pool. Logging statistics.
x'450A'	Checkpoint statistics were gathered, including statistics from the 64-bit storage pool. Latch management statistics.
x'450F'	Checkpoint statistics were gathered, including statistics from the 64-bit storage pool. Dispatcher statistics.
x'47'	A checkpoint was taken. This log record contains all the PSTs that were in the system.
x'56FA'	IMS external subsystem support recovery log record ID. Transaction-level statistics.
x'5901'	Fast Path log record. An input message was received.
x'5903'	Fast Path log record. An output message was sent.
x'5904'	Fast Path log record. Region occupancy statistics for EMH IFP users.
x'5936'	Fast Path log record. An output message was dequeued.
x'5937'	Fast Path log record. A synchronization point operation completed.
x'63'	Log session start and end. When X'02' is on in the second byte, the X'63' record represents only the deletion of a VTCB.
x'66'	A message is about to be enqueued or dequeued (applicable for 3614, FINANCE, and SLU P nodes, MSC links, or ISC sessions).
x'67D0'	This log record is a service trace record. Indicates the diagnostic record of a failed service request.

10.1.22 IMS Tools migration and coexistence

The IBM DB2 and IMS Tools enhance the performance of IMS and DB2. These tools were upgraded and enhanced to work with IMS 13. Some products require updates. For more information about requirements, contact your vendor or see Chapter 12, "Tools for IMS 13" on page 367.

For more information about these tools, including the IMS versions that they support, see the DB2 and IMS Tools for System z page at this website:

http://www.ibm.com/software/data/db2imstools

IMS Information Management Tools and IMS Version 13 compatibility

For more information about requirements for IBM IMS Tools with various IMS releases, see this website:

http://www-01.ibm.com/support/docview.wss?uid=swg21611198

IBM DB2 and IMS Tools PTF listing

The IMS Tools products are categorized by function. The information describes the minimum version and release levels of the IBM IMS Tools products that support IMS 13.

For more information about current PTFs for these tools, see the IBM DB2 and IMS Tools PTF listing at this website:

http://www-01.ibm.com/support/docview.wss?rs=434&context=SSZJXP&uid=swg27008646

10.2 IMS library updates

The IMS 13 publications remain the same as IMS 12.

10.2.1 Information Center

The Information Center was updated to include information about IMS 13. The Information Center is available at this website:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims13.doc/imshome_v13.htm

Figure 10-2 shows the IMS 13 main panel of the Information Center website.

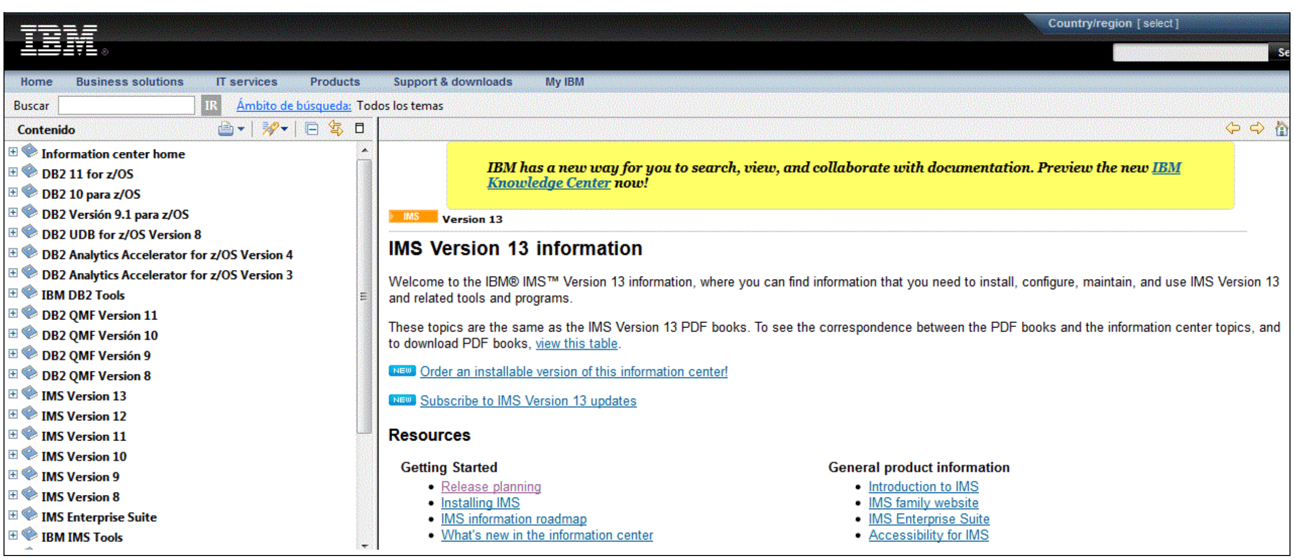


Figure 10-2 IMS 13 main panel in the Information Center website

10.2.2 IBM Knowledge Center

The new IBM Knowledge Center also was updated to include information about IMS13. The IBM Knowledge Center is available at this website:

http://www-01.ibm.com/support/knowledgecenter/SSEPH2_13.1.0/com.ibm.ims13.doc/imshome_v13.htm

Figure 10-3 shows the IMS 13 main panel of the IBM Knowledge Center website.

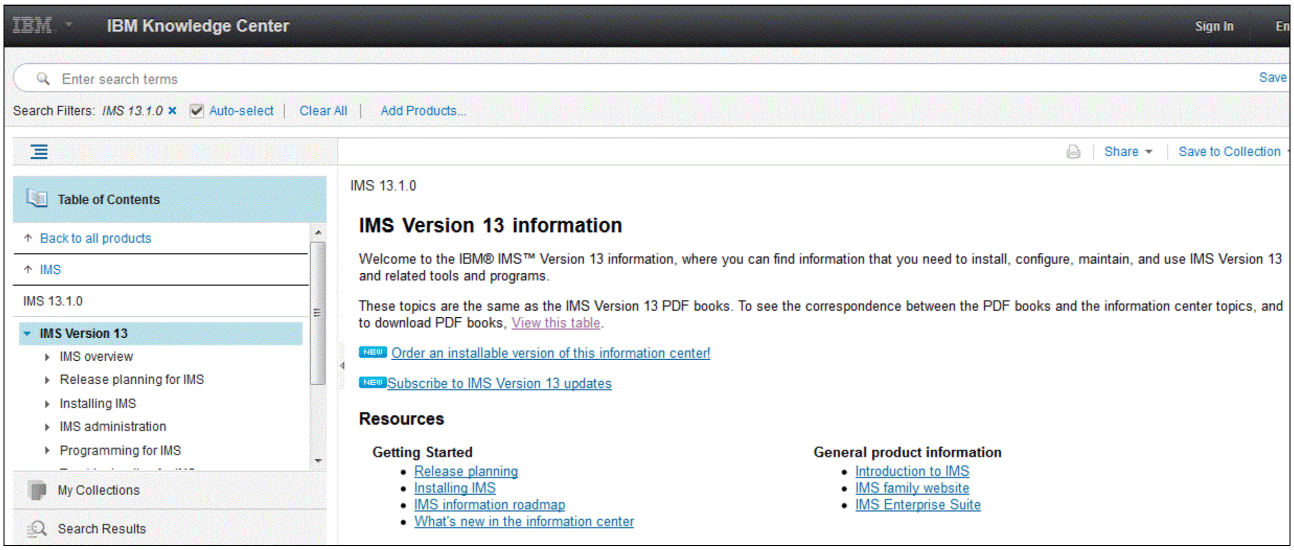


Figure 10-3 IMS 13 main panel in the IBM Knowledge Center website

10.2.3 IMS 13 information resources

The Information Center and the IBM Knowledge Center group the IMS 13 resources in several categories. Table 10-9 lists these resource categories that can vary between these centers.

Table 10-9 IMS 13 information resources in Information Center and IBM Knowledge Center

Resource	IBM Information Center	IBM Knowledge Center
Getting Started	► Release planning ► Installing IMS ► IMS information roadmap ► What's new in the Information Center	► Release planning ► Installing IMS ► IMS information roadmap ► What's new in the IBM Knowledge Center
General product information	► Introduction to IMS ► IMS Family website ► IMS Enterprise Suite ► Accessibility for IMS	► Introduction to IMS ► IMS Family website ► IMS Enterprise Suite

Resource	IBM Information Center	IBM Knowledge Center
Programming for IMS	► Application programming ► Java application development for IMS ► Application programming for SQL ► Application programming APIs ► Programming for IMS TM Resource Adapter ► System programming	► Application programming ► Java application development for IMS ► Application programming for SQL ► Application programming APIs ► Programming for IMS TM Resource Adapter ► System programming
IMS administration	► Communications and connections ► Database administration ► Operations and automation ► System administration	► Communications and connections ► Database administration ► Operations and automation ► System administration
News and events	► IMS Newsletter ► News about IMS ► Upcoming IMS events ► IMS teleconference and webcast replays	► IMS Newsletter ► News about IMS ► Upcoming IMS events ► IMS teleconference and webcast replays
Related information	► z/OS Basic Skills Information Center ► IBM Information Centers ► IBM Publications Center ► IMS Redbooks	► z/OS Basic Skills Information Center ► IBM Information Centers ► IBM Publications Center ► IMS Redbooks
IMS training	► IMS training, education, and certification	► IMS training ► IMS Mastery Certification Program
Communities	► IMS Facebook community ► IMS developerWorks® Exchange ► IMS regional user groups ► IMS YouTube channel ► IMS LinkedIn page	► IMS Facebook community ► IMS developerWorks Exchange ► IMS regional user groups ► IMS Facebook community
Support and assistance	► Send feedback about the IMS Information Center ► Search for messages and codes ► Search for technotes and APARs ► IMS Support ► Services for IMS ► IMS developerWorks site	► Search for messages and codes ► Search for technotes and APARs ► IMS Support ► IMS developerWorks site

10.3 Installation verification program

The IVP is provided by IBM to test the product installation of IMS and verify that the major functions and features are working.

The IVP process provides materials that you can use as a guide to working with your own IMS systems. The IVP process includes the following components:

- Data set allocation
- Postinstallation activities on target libraries
- System definition activities
- Supervisor call (SVC) considerations
- Authorization considerations
- IMS system preparation activities
- IMS system and application execution activities

10.3.1 Running the IMS IVP dialog

The jobs and tasks of the IVP build a sample IMS system and provide several sample applications that are used to verify specific components of IMS.

Use the IVP to verify that IMS was installed properly and that the major functions and features of IMS are working.

You should use the IVP to complete and verify the implementation and testing of IMS 13 because it is an excellent educational tool for learning how to install and customize IMS.

The IVP dialogs are documented in *IMS Version 13 Installation*, GC19-3656.

10.3.2 Phases of the IVP process

The IVP process consists of the following phases:

1. Initialization
2. Variable-gathering
3. File-tailoring
4. Execution

This section describes these phases and IVP output.

Initialization phase

The initialization phase begins when you start the IVP dialog or change an option or suboption. The IVP is driven from a set of ISPF tables, which contain information about the variables, jobs, tasks, and the sequence of those jobs and tasks you must run.

This phase has the following processes:

- The table-merge process

  A table-merge is necessary the first time you run the IVP, when you change existing environment options or suboptions that were not selected, and when the installation of service requires it.

  The table-merge process populates a set of customized tables from the master tables with the IVP variables, jobs, and tasks that are necessary for you to run the IVP that is based on your selected environment option and suboptions.

- The copy-startup-variable process

  After the table-merge process is complete or bypassed, the dialog compares the start variables with their corresponding table values.

  If the table value is different and was changed by an older copy-startup-variables process or by modifying the values of the variables in the variable-gathering phase, the table value is updated with the start value.

Variable-gathering phase

In this phase, you choose the options that are used to produce the jobs and tasks necessary in the subsequent phases of the IVP, such as file-tailoring.

File-tailoring phase

The file-tailoring phase uses the ISPF file-tailoring services to combine the variables from the variable-gathering phase with skeletons from SDFSSLIB to create members (JCL and other materials) in INSTALIB.

Execution phase

The execution phase guides you through the jobs and tasks that are necessary to complete the building and running of the IVP system that is based on options that you chose.

Only the jobs and tasks that are specific to the selections that you made during initialization are presented. The jobs and tasks are presented in the order in which they are performed.

IVP output

The IMS environments that you can select include BATCH, DBCTL, DB/DC, DB/DC with XRF, and DCCTL. Most of the major functions of IMS can be demonstrated and tested by using the IVP system. The IVP builds a viable sample IMS system in a controlled manner, which is verifiable and robust.

The IVP verifies that the IMS product and subsequent maintenance were successfully installed. It implements and verifies the z/OS and VTAM interfaces and the various functions and features that you selected.

The IVP builds, integrates, and facilitates the execution of sample IMS applications. It also assembles the database resource adapter (DRA) interface module, which is used by DBCTL and Open Database Access (ODBA).

The following functions and features are demonstrated and tested:

- Syntax Checker
- XRF
- IRLM
- Fast Path
- Shared Queues (Common Queue Server)
- High Availability Large Database (HALDB)
- IMS Connect
- IMS system restart and recovery
- Common Service Layer
- Enhanced Command Environment
- IMS DB resource adapter (previously known as the IMS JDBC Connector)
- Dynamic resource definition (DRD)
- IMS Open Database
- IMSRSC repository
- IMS Catalog

10.3.3 Starting the IVP dialog

Start the IVP dialog by running an EXEC command from an ISPF dialog or the IMS application menu.

Starting the IVP by using the EXEC command from within ISPF

You can start the IVP dialog from within ISPF by using partial syntax with a simple command or by using full syntax.

To start the IVP dialog by using partial syntax, complete the following steps:

1. Open an ISPF application dialog.

   Run the TSO EXEC command (in the ISPF panel, option 6), as shown in Figure 10-4.

   HLQ is the high-level qualifier for the IVP, system, and distribution libraries.

```
Menu  List  Mode  Functions  Utilities  Help
______________________________________________________________________________
                              ISPF Command Shell
Enter TSO or Workstation commands below:

===> EXEC 'IMS13X.SDFSCLST(DFSIXC01)' 'HLQ(IMS13X)'



Place cursor on choice and press enter to Retrieve command

=>
=>
=>
=>
```

Figure 10-4 Command to start the IVP dialog from an ISPF panel

   After the command is started, the IMS welcome panel opens, which is followed by the IBM copyright panel.

2. Press Enter to go to the IVP Environment Options panel.

Starting the IVP from the IMS Application Menu

Complete the following steps to start the IVP dialog from the IMS Application Menu:

1. Open an ISPF application dialog.
2. Start the IMS Application Menu by running the following TSO EXEC command:

   `EXEC 'IMS13X.SDFSEXEC(DFSAPPL)' 'HLQ(IMS13X)'`

   The IMS Application Menu opens, as shown in Figure 10-5 on page 282.

```
Help
------------------------------------------------------------------------------
                              IMS Application Menu
Command ===>

Select an application and press Enter.


         1   Single Point of Control (SPOC)
         2   Manage resources
         3   Reserved for future use
         4   HALDB Partition Definition Utility (PDU)
         5   Syntax Checker for IMS parameters (SC)
         6   Installation Verification Program (IVP)
         7   IVP Export Utility (IVPEX)
         8   IPCS with IMS Dump Formatter (IPCS)
         9   Abend Search and Notification (ASN)


 To exit the application, press F3.
            ------------------------------------------------
              Copyright IBM Corp. 2003. All rights reserved.
            ------------------------------------------------
```

Figure 10-5 IMS Application Menu

3. In the IMS Application Menu, select Option 6 to start the IVP.

10.3.4 Starting the IVP initialization phase

During the IVP initialization phase, you select the installation option and suboption values that the IVP uses to build customized tables of the specific jobs and tasks that need to be run.

These tables provide the input for the phases that follow. In addition, some variables are initialized in this phase in preparation for the variable-gathering phase.

Selecting the environment options

Select the options that apply to your environment. The IVP provides suboptions and tasks that are based on your choices to build a sample IMS system for installation verification.

Figure 10-6 on page 283 shows the IVP Environment Options panel. This panel is referred to as the primary option menu for the IVP dialog.

```
Help
     IVP            IVP Environment Options            Enter required field
Command ===>

Select the desired option and press Enter.


Option . .
            IVP Environments

            1. DBB - Database Management (Batch)

            2. DBC - Database Management (DBCTL)

            3. DBT - Database and Transaction Management (DB/DC)

            4. XRF - DB/DC with Extended Recovery Facility (DB/DC with XRF)

            5. DCC - Transaction Management (DCCTL)
```

Figure 10-6 IVP Environments Options panel

If you ran the IVP dialog and made a selection in the IVP Environment Options panel, the Environment Option Change Verification panel opens.

If you did not previously run the IVP dialog, the Suboption Selection panel opens.

Verifying an environment option change

When you select an environment option that you did not select before, the Environment Option Change Verification panel opens.

Figure 10-7 on page 284 shows that the new option XRF is selected and that the last selected option was DBB.

```
Help

    IVP          Option Change Verification - DBT          IMS 13.1
Command ===>


  The Option you have just chosen is not the same as the Option which was last
  active:



      XRF       - Requested Option

      DBT       - Previous  Option



  To confirm your change of Options to XRF       : Press Enter

  To return to the Option Selection menu: Press End .
```

Figure 10-7 Environment Option Change Verification panel

To verify an environment option change, review the panel contents and the requested option change.

If the requested option change is correct, press Enter to confirm your selection.

If the requested option is not correct, press End to return to the Environment Option Selection panel.

Selecting suboptions

Choose the suboptions that you want to add to your primary option selection. Suboptions specify whether you want to use IRLM, Fast Path, and other IMS functions and features.

Ensure that the corresponding FMIDs for selected suboptions are installed during IMS product installation when SMP/E is used.

Figure 10-8 on page 285 shows the IVP Sub-Options Selection panel of the IVP dialog.

```
Help

IVP                 Sub-Option Selection - DBT                  IMS 13.1
Command ===>


Select the desired Sub-Options and press ENTER

/  IRLM - Use IRLM in IVP Applications
/  FP   - Use Fast Path in IVP Applications
/  ETO  - Use Extended Terminal Option
/  CQS  - Add CQS to CSL Applications
/  RACF - Use RACF Security
/  JAVA - Use JAVA Applications and Open Database
   PRA  - Use Parallel RECON Access
/  ICON - Use IMS Connect
/  REPO - Use IMSRSC Repository
/  COUT - Use Callout Applications

  Note: Your Sub-Option selection affects the user variables, jobs, and tasks
  that will be presented.  If you later change your selection, you must redo
  the IVP Table Merge, Variable Gathering, File Tailoring, and Execution
  processes. RACF is required when Java suboption is selected.
```

Figure 10-8 IVP Sub-Option Selection panel

If you change the selections that are displayed, the Sub-Option Change Verification panel opens, as shown in Figure 10-9 on page 286.

```
Help

                  Sub-Option Change Verification - DBT
Command ===>


  The Sub-Options you have just chosen are not the same as the Sub-Options
  which were last active. If you change Sub-Options, Table Merge and the three
  Dialog Phases must be rerun from the beginning.

  From To
  Y    Y   - IRLM - Use IRLM in IVP Applications (not available for DCCTL)
  Y    Y   - FP - Use Fast Path in IVP Applications (not available for DCCTL)
  Y    Y   - ETO - Use ETO (not available for Batch and DBCTL)
  Y    Y   - CQS - Add CQS Applications (not available for Batch and DBCTL)
  Y    Y   - RACF - Use RACF Security (not available for Batch)
  Y    Y   - JAVA - Use JAVA Applications and Open Database
  N    Y   - PRA - Use Parallel RECON Access (not available for Batch)
  Y    Y   - ICON - Use IMS Connect
  Y    Y   - REPO - Use IMS Repository
  Y    Y   - COUT - Use Callout Applications

  To confirm your change of Sub-Options: Press ENTER
  To return to the Sub-Option Selection menu: Press END
```

Figure 10-9 IVP Sub-Option Change Verification panel

If you are changing the selections after you complete the table-merge, variable gathering, file-tailoring, or execution phases, you must rerun the jobs and tasks in those phases.

Requesting a table merge

After you select an environment option and suboptions, the IVP dialog gives you the option of performing a table-merge.

Complete the following steps to request a table merge:

1. In the Table Merge Request panel that is shown in Figure 10-10 on page 287, enter `1` (for YES) and press Enter. While the table merge is in progress, the Table Merge in Progress panel opens and the keyboard is locked. This panel is updated as the tables are updated.

```
Help

 IVP               Table Merge Request - DBT                IMS 13.1
Command ===>


The IVP dialog is driven from a set of ISPF tables which contain information
about the variables, JOBs, TASKs, and sequence of presentation you will need
to
perform the verifications.

Since the tables will be updated by the dialog, working copies must be made
the first time you use the dialog.

If service is applied to your IMS system, or if you decide to use the IVP
dialog to build a different environment, then either the existing copies must
be updated or new copies created.

Please indicate whether you wish to perform Table Merge/Create:

1  1. YES - Create / Update working tables from master tables.
   2. NO  - Use existing tables.
```

Figure 10-10 IVP Table Merge Request panel

2. After the table merge process completes, the Table Merge Completed panel is displayed.
3. Press Enter to continue.

Selecting an IVP phase and positioning option

Select an IVP phase and choose to start or restart from the beginning of an IVP phase or from the last known location within a phase.

Figure 10-11 on page 288 shows the IVP Phase Selection panel of the IVP dialog.

```
Help

     IVP            IVP Phase Selection - DBT            IMS 13.1
Command ===>



Select the desired Phase and positioning option and press ENTER
    A.  Variable Export Utility (Export variables to a data set)

    VG s Variable Gathering s (Define user values for variables)
    1.  VG1 Start/Restart from the beginning of the phase
    2.  VG2 Start/Restart from the last known position within the phase

    FT s File Tailoring s (Create customized INSTALIB members)
    3.  FT1 Start/Restart from the beginning of the phase
    4.  FT2 Start/Restart from the last known position within the phase
    5.  FT3 Start/Restart from the beginning of a selected step

    EX s Execution s (Run the IVP jobs)
    6.  EX1 Start/Restart from the beginning of the phase
    7.  EX2 Start/Restart from the last known position within the phase
    8.  EX3 Start/Restart from the beginning of a selected step
```

Figure 10-11 IVP Phase Selection panel

10.3.5 Gathering variables

Gathering variables involves changes to prepare the JCL and other materials that are necessary for further customization in the file-tailoring phase.

When you enter the variable gathering phase, the IVP panel displays the variables that are based on your selections in the initialization phase. These variables are used later by the file-tailoring phase to customize the IVP to your environment and to create members in the INSTALIB data set.

You can import variables from an earlier iteration of the IVP dialog by using the IVP Variable Gathering Export and Import facilities.

Exporting and importing IVP variables

IVP variables can be exported and imported between IMS releases or between different IVP dialog sessions of the same IMS release.

Use the IVP Variable Export Utility (as shown in Figure 10-12 on page 289) to copy or export a set of previously used IVP variables to a sequential data set so you can import them later.

Select this utility from the IMS Application Menu, Option 7-IVP Export Utility (IVPEX).

```
Help

                         IVP Variable Export Utility
Command ===>

Enter the following information, then press enter.

 1. Select the IVP Environment
    1. DBB - Database Management (Batch)
    2. DBC - Database Management (DBCTL)
    3. DBT - Database and Transaction Management (DB/DC)
    4. XRF - DB/DC with Extended Recovery Facility (DB/DC with XRF)
    5. DCC - Transaction Management (DCCTL)

 2. Specify the IVP High-Level Qualifier(s) of the INSTATBL data set


 3. Specify the Export data set. For a PDS, include the member name.
  If the dataset does not exist, you will be prompted to create the dataset
```

Figure 10-12 IVP Variable Export Utility panel

This data set then can be imported into the IVP tables data set of the target IVP session.

Exporting variables from one IVP session to other

Complete the following steps to export variables from one IVP session and import them to the target IVP session:

1. Start the IVP Variable Export utility by running the DFSIVPEX command from an ISPF panel. You can use the ISPF split screen capability to start the IVP Variable Export utility without exiting the IVP.
2. Run the following TSO EXEC command to start the IVP Variable Export utility in which where `qqq` is the high-level qualifier for the IMS system (SYS) libraries:

   EXEC '*qqq*.SDFSEXEC(DFSIVPEX)' 'HLQ(*qqq*)'
3. The IVP Variable Export Utility panel opens. Complete the following steps in this panel:
   a. Select the environment option. Use the same option that you selected during the Initialization phase of the IVP process.
   b. Select IVP high-level qualifier (HLQ), which identifies the IVP table data set (INSTATBL) from which you are exporting the variables.
   c. Enter the export data set name in TSO data set format.
   d. Press Enter to export the variables in the current IVP environment to the target IVP session.
4. If the export data set does not exist, the IVP Export Data Set Allocation panel opens, as shown in Figure 10-13 on page 290.

```
Help

                        IVP Export Dataset Allocation
Command ===>

Export Dataset Does not exist.

  Select an option to allocate the dataset:
     1. DSUTIL - ISPF Dataset Utility Panel (3.2)

     2. ALLOC  - Allocate using TSO allocate Command


TSO Allocate Command:
ALLOC FILE(EXPORT) DATASET(IMS13X.IVPEXPRT)   NEW CATALOG SPACE(1 1) TRACKS
R ECFM(F B) LRECL(80) BLKSIZE(0)
```

Figure 10-13 IVP Export Data Set Allocation panel

Select one of the following options to allocate the data set:

– DSUTIL

  The ISPF Utility Data Set Utility panel opens. Specify the following attributes for the export data set:

  - DSORG: Sequential or partitioned
  - RECFM: FB
  - LRECL: 80
  - BLKSIZE: Multiple of 80

– ALLOC

  Enter the name of the data set in the TSO Allocate Command field. The data set name that you specify in the panel is used to issue the TSO ALLOCATE command to allocate the data set.

  If the export data set name includes a member name, the TSO ALLOCATE command allocates a PDS data set.

Press PF3 or End to return to the IVP Variable Gathering panel.

5. If the current IVP environment does not match the environment in which the variables were exported, the IVP Import Environment Mismatch panel opens. You can choose to continue the import process or cancel it.

   The exported variables are associated with their specific IVP environment.

6. Complete the following steps to import the variables to the target IVP session from the export data set:

   a. In the Variable Gathering (LST mode) panel, run the import action command (**`Imp`**) in the action field of any variable in the panel. This command imports all of the variables from an IVP export data set.

   b. Enter the name of the export data set name in the TSO data set format. If the data set is a partitioned data set, include the member name.

Making global changes to variables

Use the export and import process during the variable gathering phase to make global changes to variables by using the ISPF editor (for example, to change the release from `12` to `13`) before you import them into a new IMS system.

10.3.6 Tailoring files

In the file-tailoring phase, the IVP uses the variables that you specified during the variable-gathering phase to prepare a customized set of IVP JCL and tasks to be stored as members of the INSTALIB data set for use in the execution phase.

The ISPF file-tailoring facility creates this input by updating and building members in the INSTALIB data set that are based on the options you choose in this phase.

Table 10-10 lists the IVP naming convention of the INSTALIB members for jobs and tasks according to the environment option that was chosen and presented as `IV_ssnt`:

Table 10-10 IVP naming convention for jobs and tasks

IV_ssnnt	Meaning	Description
_	Underscore	The environment that you selected on the first panel: 1: DBB - Batch 2: DBC - DBCTL 3: DBT - DB/DC 4: XRF - DB/DC with XRF 9: DCC - DCCTL
ss	Step	IVP step
nn	Number	The number of the item within the step. The item numbers are not guaranteed to be in ascending sequence. Service changes might disrupt the apparent sequence.
t	Item type	► J (JOB) A PDS member with the same name is placed into INSTALIB during the file-tailoring phase. Items of type J are intended to be submitted for execution. ► T (Task) Tasks represent items of work that must be prepared by the user. For some tasks, an example is provided in the INSTALIB data set. These examples are not intended for execution. ► N (Non - job) The INSTALIB data set can also contain members that support other jobs, such as CLISTs, control statements, and examples.

10.3.7 IVP jobs and tasks

The jobs and tasks that are presented in groups by the IVP dialog are determined by your choice of environment option and distribution media.

The last group that is listed, Steps Zx for index of more PDS members, does not identify jobs or tasks in the IVP process. It identifies the members of DFSSLIB and DFSISRC libraries that support the IVP process.

Figure 10-14 shows an example of the initials jobs and tasks that are presented by the IVP dialog.

```
Help

                       Execution (LST Mode) - DBT          Row 1 to 17 of 305
Command ===>                                                 Scroll ===> CSR

Action Codes: Brm Doc Edm eNt eXe Ftl spR
      JOB/Task    Step  Title..........................................
      IV3A001T    A0    NOTE - Introduction - Dialog Set-up
      IV3A301N    A3    CLIST - Offline Formatted Dump - IVP1/2/3/4
      IV3A302N    A3    CLIST - Offline Dump Formatter - BATCH
      IV3A303N    A3    CNTRL - MSDB Load Cntrl Stmts - DBFSAMD1/DBFSAMD2
      IV3C001T    C0    NOTE - Introduction - System Definition
      IV3C101J    C1    JOB  - Alloc SYSDEF Data Sets
      IV3C105J    C1    JOB  - Assembly/Bind RACF Security Exits
      IV3C201T    C2    TASK - Browse the STAGE1 Source Deck
      IV3C202J    C2    JOB  - Run SYSDEF Preprocessor
      IV3C203J    C2    JOB  - Run SYSDEF STAGE1
      IV3C301J    C3    JOB  - Run SYSDEF STAGE2      >>> SEE DESCRIPTION
      IV3C401J    C4    JOB  - Run SMP/E JCLIN
      IV3C405T    C4    TASK - Edit IMS PROCLIB Members
      IV3D001T    D0    NOTE - Introduction - z/OS and VTAM Interface
      IV3D101T    D1    XMPL - Allocate Interface Data Sets
      IV3D200T    D2    XMPL - Update JESx Procedure
      IV3D201T    D2    XMPL - Update IEAAPFxx or PROGxx - Authorized DSN
```

Figure 10-14 Execution phase 6 of the initial IMS 13 IVP jobs and tasks

You can print more documentation for the IVP jobs, tasks, and variables by using the DOC action during the file-tailoring phase or the execution phase of the IVP dialog.

Use the IVP dialog to obtain current information regarding IVP jobs and tasks. In these lists, the jobs and tasks are presented in the same sequence that is used by the IVP dialog.

10.3.8 Running IVP tailored jobs and tasks

The IVP provides a set of example JCL that you can tailor to your environment in the variable gathering phase. It is normal to run through each of the steps sequentially.

This process builds your IVP system and runs each function that you selected on the Sub-Option Selection panel (see Figure 10-8 on page 285).

To run the IVP jobs and tasks, complete the following steps:

1. In the IVP Phase Selection panel, select option **6**, **7**, or **8**. Each selection within a phase provides a different positioning option.
2. Open each job or task. To view the instructions for each job and task, use the **ENT** action command.

   For IVP jobs, you can browse, edit, or submit the job. Some items are nonexecutable examples, but the browse and edit actions are available to create an executable version of nonexecutable items.

For IVP tasks, you are provided a scrollable description to assist you in performing the task.

3. Press End or PF3 when you are done.
4. Press Enter again if you completed the execution of all jobs and tasks, or press End to save your work if you want to complete the execution phase later.

10.3.9 IMS 13 IVP enhancements for ISC

In IMS 13, the IVP is enhanced to support the ISC over TCP/IP function. For more information, see 3.4, “Intersystem Communication over TCP/IP” on page 84.

The IVP is enhanced to support the new IMS Connect RMTCICS, ISC, and TCPIP configuration statements in the IMS Connect HWSCFGxx PROCLIB member for ISC links between IMS Connect and CICS.

Repository server variables

Tables 10-11 lists four new variables that are defined in the repository server configuration.

Table 10-11 IVP repository server variables

Variable	Function	Description
IXUDNAME	Descriptor name	Specifies the name of the logon descriptor.
IXUCONNM	IMS Connect name	Specifies the local IMS Connect instance.
IXUOUTSZ	Output buffer size	Specifies the size of the output buffer to be used. This keyword is the same as the OUTBUF= keyword on the TERMINAL macro.
IXUSEGSZ	Segment size	Specifies the segment size. The acceptable values are 256 - 32000 bytes (default is 256 bytes). For more information about calculating segment size, see the description of the SEGSIZE= keyword in TERMINAL macro at IMS system definition.

Proclib member HWSCFGxx ISC definitions

Variables were added that are used to build proclib member HWSCFGxx for ISC link.

Figure 10-15 on page 294 shows the job IV_E302J that appears in the IVP dialog and adds control statements to the IMS.PROCLIB data set.

```
Help

                       Execution (LST Mode) - DBT         Row 38 to 54 of 305
Command ===>                                                 Scroll ===> CSR

Action Codes: Brm Doc Edm eNt eXe Ftl spR
      JOB/Task    Step  Title...........................................
      IV3E001T    E0    NOTE - Introduction - Build IVP Appl / System
      IV3E101J    E1    JOB  - Allocate Data Sets
      IV3E201J    E2    JOB  - DBDGENs
      IV3E202J    E2    JOB  - PSBGENs
      IV3E203J    E2    JOB  - ACBGEN
      IV3E204J    E2    JOB  - MFS Language Utility
      IV3E206J    E2    JOB  - Assembly/Bind Applications
      IV3E207J    E2    JOB  - Assembly/Bind Install. Default Block
      IV3E301J    E3    JOB  - Create Dynamic Allocation Members
      IV3E302J    E3    JOB  - Add Control Statements to IMS.PROCLIB
      IV3E303J    E3    JOB  - Add SCI/OM/RM Members to IMS.PROCLIB
      IV3E304J    E3    JOB  - Add CQS Members to IMS.PROCLIB
      IV3E305J    E3    JOB  - Define EXEC PARM Defaults
      IV3E306T    E3    TASK - Syntax Checker Sample
      IV3E307T    E3    TASK - Define z/OS Policies
      IV3E308J    E3    JOB  - Define DRA Start-up Table
      IV3E309J    E3    JOB  - Verify TCO Scripts
```

Figure 10-15 Job IV3E302J in IVP dialog adds control statements to IMS.PROCLIB

Proclib member DFSDSCTy definitions

Job IVPE302J was updated to add the configuration member DFSDSCTy (ETO User Descriptor Table) into the IVP PROCLIB member.

IV3E302J job step with ISC link definitions

Figure 10-16 on page 295 shows IV3E302J job step that add the HWSCFGxx proclib member with the ISC link definitions to the IMS Connect configuration.

```
 BROWSE    IMS13X.INSTALIB(IV3E302J) - 01.00         Line 00000340 Col 001 080
 Command ===>                                                  Scroll ===> CSR
//*
//* THE FOLLOWING STEP ADDS THE HWSCFGIV MEMBER TO IMS PROCLIB.
//* THIS MEMBER DEFINES IMS CONNECT CONFIGURATION.
//*
//HWSCFGIV EXEC PROC=PROCUPDT,MBR=HWSCFGIV
//P.SYSUT1 DD *
HWS=(ID=IVP13HWS,XIBAREA=100,RACF=N)
ADAPTER=(XML=Y)
TCPIP=(CICSPORT=(ID=1111,KEEPAV=0),
EXIT=(HWSSMPL1),HOSTNAME=TCPIP,IPV6=Y,
PORTID=(9999))
ISC=(ID=CICSA1,NODE=ISCTCP1,
     LCLIMS=IVP1,
     IMSPLEX=(MEMBER=HWS1,TMEMBER=PLEX1),
     RMTCICS=CICS1A,CICSAPPL=CICS1,
     CICSPORT=1111,CICSNETID=CICSNTI1)
RMTCICS=(ID=CICS1A,HOSTNAME=TCPIP.HOSTNAME,PORT=4444)
DATASTORE=(GROUP=IVPXCFGN,ID=IVPDSNM,MEMBER=IVP13HWS,
TMEMBER=IVP1)
/*
```

Figure 10-16 IV3E302J job step adding ISC link definitions to IMS Connect configuration

10.4 Syntax Checker

The Syntax Checker is an ISPF application that helps you define, verify, and validate parameters and their values in the members of the IMS PROCLIB data set.

Use the IMS Syntax Checker to avoid typographical and syntactical errors when you are modifying the parameter values for the IMS PROCLIB data set members.

The Syntax Checker includes the following features:

- Reads in the IMS PROCLIB data set member.
- Displays the parameters and values.
- Allows modification of the parameter values.
- Helps migrate supported IMS PROCLIB data set members from one version of IMS to another.
- Displays default values for parameters.
- Verifies the validity of parameters and values.
- Saves changes back to the IMS PROCLIB data set member.

The Syntax Checker provides an ISPF panel in which you can add or change parameter values in the members.

Online help is available at the parameter level and assists in moving to a new IMS release by identifying new parameters and any obsolete parameters from the previous release. It saves the parameters to appropriate members of the IMS PROCLIB data set in the correct format. The next time the control region is started, it uses the new values.

10.4.1 Starting the Syntax Checker

You can start the IMS Syntax Checker by using of the following methods, an EXEC command or from the IMS Application Menu:

- Start the Syntax Checker by issuing the following command in ISPF option 6:

  `EXEC 'HLQ.SDFSEXEC(DFSSCSRT)' 'hlq(HLQ)'`

- You can also start the Syntax Checker from the IMS Application Menu.

  The IMS Application Menu provides a common interface to IBM supplied IMS applications that run on TSO and ISPF. The Syntax Checker remains running until you exit it manually.

  Figure 10-17 shows this selection in the IMS Application Menu.

```
Help

                             IMS Application Menu
Command ===> 5

Select an application and press Enter.


         1   Single Point of Control (SPOC)
         2   Manage resources
         3   Reserved for future use
         4   HALDB Partition Definition Utility (PDU)
         5   Syntax Checker for IMS parameters (SC)
         6   Installation Verification Program (IVP)
         7   IVP Export Utility (IVPEX)
         8   IPCS with IMS Dump Formatter (IPCS)
         9   Abend Search and Notification (ASN)


 To exit the application, press F3.
          ___________________________________________________
          _ Copyright IBM Corp. 2003. All rights reserved. _
          ___________________________________________________
```

Figure 10-17 IMS Application Menu

10.4.2 Using the Syntax Checker

After you start Syntax Checker, the IMS Parameter Syntax Checker panel that is shown in Figure 10-18 on page 297 is displayed. Here, you can enter the PROCLIB data set name and the name of the member to be processed.

If you do not enter the member name, a member list is displayed. Select the member you want to process from the list, as shown in Figure 10-18 on page 297.

```
 File  Help

                         IMS Parameter Syntax Checker
 Command ===>

 Enter the name of the IMS proclib dataset and press enter.


  ISPF Library:
     Project  . .
     Group  . . .
     Type . . . .
     Member . . .           (Blank for member list)



  Other Partitioned Data Set:
     Data Set Name  . . 'IMS13X.PROCLIB(DFSPBIV1)'
     Volume Serial  . .           (If not cataloged)
```

Figure 10-18 Syntax Checker entry panel

Table 10-12 lists IMS PROCLIB data set members that are processed by the Syntax Checker.

Table 10-12 IMS PROCLIB members that are processed by Syntax Checker

Member	Description
BPE config	BPE execution environment settings, such as tracing, language, and statistics time.
CQSIPxxx	Specify parameters that are related to initialization of the CQS address space.
CQSSGxxx	Define global CQS parameters that are related to one or more coupling facility structures.
CQSSLxxx	Define local CQS parameters that are related to one or more coupling facility structures.
CSLDCxxx	Define the data store connections between one or more ODBM instances and one or more IMS systems.
CSLDIxxx	Specify parameters that initialize the ODBM address space.
CSLOIxxx	Specify parameters that initialize the Operations Manager (OM) address space.
CSLRIxxx	Specify parameters that initialize Resource Manager (RM) address space.
CSLSIxxx	Specify parameters that are related to initialization of the SCI address space.
DFSCGxxx	Specify parameters that are related to the Common Service Layer (CSL), including the OM, RM, and the Structured Call Interface (SCI).
DFSDCxxx	Define data communication options.
DFSDFxxx	Consolidates the specification of processing options for many different IMS components and functions.
DFSPBxxx	Specify execution parameters for the DBCTL, DCCTL, and DB/DC control regions.

Member	Description
DFSSQxxx	Specify parameters that are related to the shared message queues and the CQS address space in DB/DC and DCCTL environments.
DSPBIxxx	Specify parameters that initialize the DBRC address space.
FRPCFG	Define the Repository Server (RS) configuration parameters that relate to performance, communications, and security. FRPCFG also identifies the names of the RS catalog repository data sets.
HWSCFGxx	Specify environmental settings for IMS Connect.

When you press Enter from the main panel, Syntax Checker reads the input file and tries to determine the IMS release and type of control region.

If Syntax Checker cannot determine this information, one of the following entry panels opens:

- IMS Release and Control Region Type entry
- IMS Release entry

If Syntax Checker can determine the information it requires from comments in the member, the Syntax Checker Keyword Display panel is shown. See the Keyword Display panel that is shown in Figure 10-21 on page 300.

10.4.3 IMS Release and Control Region Type entry panel

The IMS Release and Control Region Type entry panel that is shown in Figure 10-19 provides Syntax Checker with IMS release and control region information that is required to process the member correctly.

```
File  Help

                         IMS Parameter Syntax Checker
Command ===>

Enter the following information and press enter.

IMS Release  . . . . . . . . 1  1. IMS 13.1
                                2. IMS 12.1
                                3. IMS 11.1


Type of Control Region . . . 3  1. DBCTL Control Region
                                2. DCCTL Control Region
                                3. DB/DC Control Region
                                4. FDBR Region
                                5. DLI/DBB Batch
```

Figure 10-19 IMS Release and Control Region Type entry panel

When Syntax Checker saves the member, it adds comment lines to the top of the member that is saving this information. The next time Syntax Checker processes the member, this panel does not display.

After you enter the data and press Enter, the Syntax Checker Keyword Display panel opens.

10.4.4 IMS Release panel

The IMS Release panel that is shown in Figure 10-20 prompts you to provide the Syntax Checker with IMS Release information that is required to process the member correctly.

```
File  Help

                         IMS Syntax Checker
Command ===>

Enter the following information and press enter.

IMS Release  . . . . . . . .    1. IMS 13.1
                                2. IMS 12.1
                                3. IMS 11.1
```

Figure 10-20 IMS Release panel

After you enter the data and press Enter, the Syntax Checker Keyword Display panel opens.

10.4.5 Keyword Display panel

The Keyword Display panel of the IMS Syntax Checker that is shown in Figure 10-21 on page 300 displays the keywords and their values and indicates whether syntax errors exist.

```
File  Edit  View  Help

                   IMS 13.1 Parameters for DB/DC
Command ===>

Press enter (without other input) to check for errors.

Data Set Name  . . : IMS13X.PROCLIB(DFSPBIV1)
IMS Release  . . . : 13.1

  Sel Codes: C = Comment D = Delete I = Insert P = Process / = Select


 Sel Keyword    Value              Description                        More:  +
  _  UHASH = ,                     Unknown Keyword
  _  ALOT = 60                     ETO Auto Logon Off Time
  _  AOIP = ,                      AOI Pool Upper Limit
  _  AOIS = ,                      ICMD Security Option
  _  APPC = ,                      Activate APPC/IMS (Y|N)
  _  APPLID1 = SCSIMSAL            VTAM Applid of Active IMS System
  _  AP ____________________________________________________________
  _  AP  Member DFSPBIV1 will be processed under IMS 13.1 (DB/DC )
  _  AR ____________________________________________________________
  _  ARMRST = N                    Allow MVS ARM to Restart (Y|N)
  _  ASOT = 60                     ETO Auto Signoff Time
  _  AUTO = N                      Automatic Restart Desired (Y|N)
  _  BSIZ = 02048                  Data Base Buffer Size
  _  CIOP = ,                      CIOP Pool Upper Limit
  _  CRC = ,                       Command Recognition Character
  _  CSAPSB = 12                   DLISAS: CSA PSB Pool Size
  _  DBBF = 00050                  Number of Database Buffers
  _  DBFX = 00010                  Num. DB Buffs available at FP Reg Start
  _  DBRCNM = IVP13RC1             DBRC Problib Member Name
  _  DBWP = 024                    Database Work Pool Size
  _  DLINM = IVP13DL1              DL/I Proclib Member Name
  _  DLIPSB = 40                   DLISAS: DLI PSB Pool size
  _  DMB = 048                     DMB Pool Size
  _  DSCT = I                      ETO User Descriptor Table(DFSDSCTX)
  _  EMHB = ,                      EMHB Pool Upper Limit
  _  EMHL = 256                    Size of EMH Buffer in bytes
  _  EPCB = 0012                   EPCB Pool Size
  _  ETO = Y                       Extended Terminal Option (y|n|m)
```

Figure 10-21 Keyword Display panel

If you select **Display new** from the View menu, the Syntax Checker lists only those parameters that are new in the IMS version you selected.

To display the default values for parameters, press F6; press the key again to toggle between displaying and not displaying defaults. The default values are displayed in the description field of the parameter.

After the member is modified, press the Enter key without making any other modifications. Syntax-value checking is performed.

If errors exist, the first keyword with an error moves to the top of the display and an error message is displayed.

When all errors are resolved and you modified the member as required, you can save the member to the originally selected PROCLIB and member or to a different PROCLIB and member.

10.4.6 Exiting the Syntax Checker

When you exit the Syntax Checker, it prompts you to save any unsaved changes.

10.4.7 IMS 13 Syntax Checker enhancements

When you use the Syntax Checker to check parameters for earlier releases of IMS, you must verify that the correct release number is displayed.

The IMS 13 Syntax Checker supports IMS 11, IMS 12, and IMS 13.

The Syntax Checker is enhanced in IMS 13 to support the RCLASS and SECCNT initialization parameters.

10.5 Installation and migration tasks

In this section, we provide an overview of the tasks that are needed to migrate to IMS 13.

10.5.1 Migration considerations

This section describes practices that are useful for a general migration of current IMS installations regarding the following tasks:

- Preparation
- Installation
- Validation

Installation documentation is provided with CBPDO, ServerPac, and so on.

Preparation tasks

Perform the following tasks before the migration:

- Ask for all installations of new products the Preventive Service Planning (PSP) bucket. The Program Directory for the products should be reviewed before an IMS migration is started.
- Contact IBM Software Support for current installation, migration, and problem-resolution information. Ask for the PSP upgrade for IMS 13 that is IMS1300.
- Before you install IMS 13, check with your IBM Support Center or use Information/Access or Service Link to determine whether more PSP information is available of which you must be aware.

  An alternative to the use of PSP is to use fix category hold data (see FIXCAT categories in 10.1.9, “Prerequisite maintenance” on page 269).

Documentation review

Review the following documentation:

- Read *Program Directory for Information Management System Transaction and Database Servers V13.0*, GI10-8914, for the most current hardware requirements, software requirements, prerequisites, and installation information.
- Review the "IMS Installation Overview" chapter in *IMS Version 13 Installation*, GC19-3656.
- Review the service that was applied to your current system. Determine whether critical service was included in the new IMS release. If not, acquire the appropriate service for the new IMS release. Again, the new fix categories process can help you with this task.
- Review the functions and enhancements in *IMS Version 13 Release Planning*, GC19-3658. In particular, review the changes to the following areas:
  - SMP/E, distribution, and system data sets
  - System definition macros
  - Log records
  - RECON records
  - Exit routines
  - Cataloged procedures
  - Control statement members in the PROCLIB data set
  - Utilities
  - Operator commands
  - Operating procedures
  - Messages and abend codes
- Review the "z/OS interface considerations" chapter in *IMS Version 13 System Administration*, SC19-3659, which explains Supervisor Call and the SYS1.PARMLIB updates.

Maintenance installation

Complete the following steps:

1. Install prerequisite software and maintenance.
2. Determine the availability of updates to all tools, aids, and related products. They might need to be upgraded or new releases for use with IMS 13. For more information about IBM IMS Tools, see this website:

   http://www.ibm.com/support/entry/portal/Installation/Software/Information_Management/IMS_Tools
3. Apply coexistence SPEs to your existing lower-level IMS systems that operate with IMS 13.
4. Examine Hardware Data Compression (HDC) dictionaries to determine whether they incorporate IMS versions that are now out of service.

   Although rebinding dictionaries is not required when you are migrating to a new version of IMS, a gradual refresh of these dictionaries to a current release is a good practice.
5. Evaluate and update IMS exit routines for use with IMS 13. It is generally advisable to reassemble all assembler exits when you are migrating to a new IMS release to ensure that the latest macro changes are included.

   If you use a RECON I/O Exit Routine (DSPCEXT0), examine it for required changes because of the change in RECON records.

   There is a new parameter available for use that enables sharing of storage between the security-related exits DFSCTSE0, DFSCTRN0, and DFSCSGN0. If you choose to use this new parameter, these exits must be updated to use it.

The DFSMSCE0 exit must be reassembled for use with IMS 13.

6. All locally modified IMS Connect exit routines must be reassembled when you are migrating IMS Connect to IMS 13.

   In IMS 12 and later, IMS ships load modules for the user exit routines HWSUNIT0, HWSJAVA0, HWSSMPL0, HWSSMPL1 in addition to the source. Previous versions only supplied the source code.

Installation tasks

To install IMS 13, complete the following tasks:

1. Install IMS 13 by using the SMP/E installation process.
2. Use CBPDO or ServerPac.

   CBPDO contains the base product and provides service for the products that are included with the product order. The service includes all PTFs that are available within one week of order fulfillment. All PTFs are identified by one or more SOURCEIDs, including PUTyymm, RSUyymm, SMCREC, and SMCCOR.

   Install the ServerPac package on your system and complete the installation of the software it includes through the CustomPac Installation dialog. ServerPac allocates, catalogs, and loads all the data sets. It also sets up the SMP/E environment, supplies a job to update PARMLIB members, and directs you to start the IVP.
3. Validate your system definition stage1 source. Consider merging some elements from the IVP source with your source. The IVP Variable Export utility makes the migration of IVP variables between releases easier.
4. Run the IMS IVP after the installation of a new IMS system. Running the IVP is optional, but recommended. All required installation tasks are done outside the IVP. The IVP verifies that the installation is correct, and can be used sporadically afterward.
5. Install the system prerequisites and your new IMS system, including the pre-generation service. The complete set of IMS 13 modules that are necessary for execution are built by a combination of SMP/E processing and running a stage1 or stage2 ALL system definition process.

   System definition is required as with previous IMS releases. Most system definition statements from previous IMS releases are compatible with IMS 13.

   If necessary, you can use the SMP/E `GENERATE` command to generate the JCL for jobs that build the modules that are not built during the system definition process.
6. Install required service that was not included in the pre-generation service.
7. Install any required updates to IBM IMS and DB2 tools, aids, and related products.
8. Install in z/OS system the IMS 13 Type 2 and Type 4 SVC modules that were created by IMS system definition process.

   > **Note:** A z/OS IPL is not required. They can be installed by running the `DFSUSVC0` command and specifying `SVCTYPE=(2,4)`. The utility must be rerun after every z/OS IPL, which permanently installs them at the next scheduled z/OS IPL.

9. Review Chapter 5 of *IMS Version 13 System Administration*, SC19-3659.

   IMS 13 installs the IMS abend formatting module DFSAFMX0 and uses it dynamically. No user setup is required and you do not need to install module DFSAFMD0 on the host z/OS system if you are running only versions IMS 11 or later.

10. Perform any DB recoveries that include Extended Error Queue Elements (EEQEs) by using the `/DIS DB EEQE` command.
11. Upgrade the RECON data set by using the `CHANGE.RECON UPGRADE` command with the IMS 13 DBRC utility.

    You should apply DBRC coexistence SPEs to your IMS 12 or IMS 11 systems before you upgrade your RECONs to IMS 13. This task is required for the IMS 12 or IMS 11 systems to use the RECONs after the upgrade.
12. Build application control blocks with a full ACBGEN. It is required for use with the online system or any batch DBB jobs.

Restriction: It is not possible to run IMS 13 with an ACB library built by using IMS 11 or IMS 12. It also is not possible to run IMS 11 or IMS 12 with an IMS 13 ACB library. The IMS system fails to load DBDs and PSBs because they are not compatible.

Validation tasks

Perform the following validation tasks:

1. Validate possible changes to users' cataloged procedures or jobs. Ensure that database resource adapter (DRA) client address spaces (CICS,DB2 SPAS, and so on) point to correct SDFSRESL (DRA and JCL).

   The version of the IMS DRA modules that are used by a DRA client must be the same version as the IMS with which the DRA client is communicating.
2. Validate user-created members of the PROCLIB data set.

   Use IMS 13 Syntax Checker to convert members from IMS 12 or IMS 11 where appropriate.
3. Validate, reassemble, and rebind exit routines, especially IMS Connect exit routines and code that uses IMS control blocks, such as database randomizers. Do not forget about DFSIDEF0 (IMS Installation Defaults Block).

   Rework user modifications (USERMODS), as necessary.
4. Validate, reassemble, and rebind user programs that process log records. Some log record formats changed.
5. Ensure that appropriate Dynamic Allocation members and exits are available to the new environment. Compare members in current SDFSRESL to those in new SDFSRESL.
6. Validate and update operating procedures regarding recovery, backup, and restart procedures.
7. Review the various execution parameters in the DFSPBxxx member of the PROCLIB data set that can affect performance and migration. You can also use Syntax Checker to validate the values for the DFSPBxxx parameters.

   Review and set the appropriate values for the `AOIP`, `CMDP`, `DYNP`, `EMHB`, `FPWP`, `HIOP`, `LUMC`, and `LUMP` parameters to specify an upper limit on the amount of storage a pool can acquire.
8. When MSC is used to connect IMS systems with different releases, consider all message types and the prefix sizes that accompany them. Such message types include Intersystem Communication (ISC), Advanced Program-to-program Communication (APPC), and Open Transaction Manager Access (OTMA).

   When message queue data sets are used, ensure that the MSGQ LRECL and block sizes are identical across all IMS MSC systems.

A problem can occur when an IMS system is migrated to a new release that uses messages with larger prefix sizes and new prefix segment types. Messages that contain these new and larger prefixes are sent to an earlier release of IMS and they might not fit in the message queues of the earlier release of IMS.

9. Consider other products that can be affected by migration. Any product that depends on the format and contents of the IMS log or the RECON data set is potentially affected. Examples include the following products or utilities:
   - IMS Statistical Analysis utility
   - IMS Fast Path Log Analysis utility
   - IMS Log Transaction Analysis utility
   - IMS MSC Log Merge utility
   - CICS
   - Non IBM products, including user modifications
10. Stop your pre-version 13 system.
11. Ensure archive runs successfully.
12. Cold start your IMS 13 by using the `/NRE CHKPT 0 FORMAT ALL` command.
13. Test your IMS 13 system:
    - Consider changes in IMS commands
    - Monitor changes in storage utilization (CSA/ECSA/PRIVATE/EPRIVATE)
    - Prepare and test a fallback plan

Restriction: Extended checkpoints cannot be used to restart applications across different IMS versions.

10.5.2 Discontinued support in IMS

Support is discontinued for various utilities, exit routines, and functions, which are described in this section.

IMS SMU

IMS 9 was the last version of IMS to support the security management utility (SMU). IMS supplied SMU-to-RACF utilities to assist in the migration from SMU to RACF.

Note: These utilities are not supplied after IMS 11. Users of SMU should migrate from SMU to RACF or an equivalent product while on IMS 9 or an earlier version of IMS.

IMS security

IMS 12 is the last release to support the SECURITY macro in system definition.

Use the initialization parameters to define RCLASS and SECCNT in IMS 13.

This capability was retrofitted to IMS 11 by PM48203 (UK74050 V11 SPE) and IMS 12 by PM48204 (UK74051 V12 SPE). If the RCLASS parameter is specified in DFSPBxxx/DFSDCxxx proclib members, you also need PM72199/UK82616 (IMS 11) and PM73558/UK82617 (IMS 12).

The following IMS security exits are no longer in IMS 13 nucleus. They are now stand-alone in 31 bit private:

- DFSCSGN0 (now called at IMS initialization)
- DFSCTRN0
- DFSCTSE0

IMS Enterprise Suite

IMS Enterprise Suite Version 2.1 is the last release of IMS Enterprise Suite that includes the DLIModel utility plug-in.

To use the most current enhancements, customers that use the IMS Enterprise Suite DLIModel utility plug-in must migrate to the use of the IMS Enterprise Suite Explorer for Development instead.

Although IMS Explorer for Development includes enhancements to most IMS Enterprise Suite DLIModel utility plug-in functions, it does not provide support for IMS XML DB or IMS database web services.

Continue to use the IMS Enterprise Suite Version 2.1 DLIModel utility plug-in only if these functions are required.

JCA resource adapter

The JCA 1.0 resource adapter (one of the Java connectors in the IMS DB distributed resource adapter) is stabilized and is no longer enhanced.

Because the IMS Universal drivers are built on industry standards and open specifications, and provide more flexibility and improved support for connectivity, data access methods, and transaction processing options, use the IMS Universal DB resource adapter.

IMS Connect SSL function

IMS 13 is the last version of IMS to support the IMS Connect SSL function.

Migrate to the use of IBM z/OS Communications Server Application Transparent Transport Layer Security (AT-TLS) to set up Secure Socket Layer (SSL) on TCP/IP connections to IMS Connect.

MFS web enablement

IMS 13 is the last release to support IMS MFS web enablement. Support for MFS web enablement in IMS 13 extends only to current users of the function.

Migrate to IBM Rational Host Access Transformation Services (HATS).

IMS Classic Java APIs

IMS 13 is the last release to support IMS Classic Java APIs.

Migrate to the IMS Universal JDBC Driver.

Enterprise Workload Manager

Support for Enterprise Workload Manager (EWLM) is withdrawn:

However, transition assistance for EWLM 2.1 customers is available. Also, Tivoli tools offer workload automation and monitoring solutions.

Knowledge Based Log Analysis

Knowledge Based Log Analysis (KBLA) is not supported after IMS 11. Customers that are using this function must migrate to use the following IMS-provided analysis utilities and reports:

- Fast Path Log Analysis utility (DBFULTA0)
- File Select and Formatting Print utility (DFSERA10)
- IMS Monitor Report Print utility (DFSUTR20)
- Log Transaction Analysis utility (DFSILTA0)
- Offline Dump Formatter utility (DFSOFMD0)
- Statistical Analysis utility (DFSISTS0)

The IBM IMS Tools, IMS Performance Analyzer, IMS Problem Investigator, and IBM Transaction Analysis Workbench for z/OS provide capabilities to analyze logs. They also provide capabilities that are similar to those of KBLA and many more extensive analysis capabilities.

10.5.3 Fallback considerations

IMS does not support downward compatibility in any major function between releases. A system might fallback to a previous release after upgrade.

Consider the following steps when you are preparing your migration fallback plan. This information is intended as a guide to understanding fallback inhibitors and is not considered complete.

For each IMS that you are falling back, complete the following steps:

1. Ensure that the status of all databases that are updated by IMS 13 is correct.

   Establish a new recovery point for these databases by image copying them before allowing updates in the fallback IMS release.

   Database recoveries use latest IMS version utilities.

2. Resolve DBRC issues. For more information, see 8.7.4, “DBRC fallback considerations”.

   Ensure that you have the correct DBRC Coexistence PTF applied to the older IMSs, UK80026 (IMS 11), and UK80027 (IMS 12), because upgraded RECONs are used.

3. Shut down IMS 13.

4. Ensure archive runs successfully.

5. Install the old version of IMS.

   You must rebuild your ACB library at the IMS version to which you are falling back.

   Always use ACBLIB built at the level of the running system.

6. Cold start IMS by using the `/NRE CHKPT 0 FORMAT ALL` command.

   You can use the IBM IMS Queue Control Facility for z/OS (QCF) to requeue IMS 13 messages to IMS 11 or IMS 12 message queues.

10.5.4 DBRC fallback considerations

Certain steps must be taken to revert DBRC to an IMS 11 or IMS 12 level.

For more information, see the fallback considerations that are available at this website:

http://www-01.ibm.com/support/knowledgecenter/SSEPH2_13.1.0/com.ibm.ims13.doc.rpg/ims_mig_fallback_over.htm

Base Primitive Environment-based DBRC

If you started to use Base Primitive Environment (BPE)-based DBRC and must fall back to IMS 12, there is no fallback. Complete the following steps:

1. Shut down the IMS control region that is associated with the BPE-based DBRC address space.
2. Modify the DBRC procedure to use JCL appropriate for a non-BPE-based DBRC region.
3. Restart IMS with DBRCNM referring to the non-BPE DBRC region start JCL.

You might have to revert either or both of the following exits to older versions if you changed these to not be compatible with the old calling interface:

- DSPCEXT0
- DSPDCAX0

Database Change Accumulation Utility (DFSUCUM0)

If you fall back from IMS 13 and you have change-accumulation data sets that were created by IMS 13, the logs in these data sets are not recoverable because the older utilities cannot process them.

Invalidate the IMS 13 change-accumulation data sets by running an image copy of the affected databases at the older level.

Lowering the minimum version value

Do not change **MINVERS** to `13.1` until you are sure that you do not have to fall back to a lower IMS version. However, if such a fallback becomes necessary, you can reduce **MINVERS** by completing the following steps:

1. Shut down all IMS 13 subsystems.

   Confirm that IMS 13 subsystem records were removed from the RECON data set. Run the **LIST.SUBSYS** command to see the subsystem records in the RECON data set. If IMS 13 subsystems remain in the RECON, take the necessary recover actions.

2. Determine the status of your databases.

   If the databases need recovery with IMS 13 logs, perform those recoveries with IMS 13. If the databases need recoveries to a time before the IMS 13 updates, perform those recoveries with the IMS 13 or the older version.

3. Use the following commands to delete all IMS 13 log records and all the allocation records on those logs from the RECON data set:

   `DELETE.LOG`

   `DELETE.ALLOC`

4. Reset the **MINVERS** value by running a **CHANGE.RECON MINVERS** command by using IMS 13. If you receive message DSP1205E (meaning that the database quiesce flags are active), use the **CHANGE.DB** or **CHANGE.DBDS** command to turn off the flags. After the flags are turned off, reissue the **CHANGE.RECON MINVERS** command.

5. Establish new recovery points for all databases that are updated by IMS 13 by taking image copies of those databases by using the older version of IMS.

For more information about the DBRC commands, see *IMS Version 13 Commands, Volume 3: IMS Component and z/OS Commands*, SC19-3650.

10.6 Review of migration considerations

When you are migrating to a more recent release of IMS, review the following release planning guides that span your specific migration path:

- *IMS Version 13 Release Planning*, GC19-3658
- *IMS Version 12 Release Planning*, GC19-3019
- *IMS Version 11 Release Planning*, GC19-2442

10.6.1 IMS SVCs

The following SVCs are used by IMS.

Type-2 DFSVC200 (IGCxxx)

This SVC is compatible with earlier versions for two levels and can use the version from the highest level of IMS 11 or 12. Supervisor call number is specified in SYSGEN:

- SVCNO keyword on IMSCTF macro
- DFSPBxxx member

Type-4 DSP00MVS (IGC00xxx)

This SVC is compatible with earlier versions for two levels and can use the one from the highest level of IMS 11 or 12. The SVC number is specified in SYSGEN (SVCNO keyword on IMSCTF macro).

DFSUSVC0 utility

The IMS DFSUSVC0 utility can be used to update the IMS type-2 and IMS type-4 SVCs dynamically. It loads SVC in CSA/ECSA and updates z/OS SVCTABLE to point to new module. It lasts for the life of the IPL.

In the type-2 SVC processing, it checks the IMS SVC directory table to determine whether there are any IMS tasks or jobs that are running. If jobs are running, the request to update T2 SVC fails.

In the type-4 SVC processing, it loads a new copy in ECSA each time DFSUSVC0 run requesting T4 SVC to be updated.

Figure 10-22 shows an example of the JCL that is running DFSUSVC0.

```
//STEP1     EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(2,4)'
//*STEP1 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(2)'
//*STEP1 EXEC PGM=DFSUSVC0,PARM='SVCTYPE=(4)'
//STEPLIB   DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB  DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121)
```

Figure 10-22 Example JCL for running DFSUSVC0 utility

10.6.2 Migration with DRD

Installations that implemented DRD in IMS 11 or IMS 12 can carry their DRD definitions forward to IMS 13 by using the RDDS.

When IMS 13 is cold started, it can import its database, program, transaction, and routing code definitions from an RDDS that was created with IMS 12 or IMS 11.

Similarly, for a fall back from IMS 13 to IMS 12 or IMS 11, you can cold start the IMS 12 or IMS 11 system and import definitions from an RDDS that was created by IMS 13.

For both migration and fall back, the IMSID must be the same as that used by the system that created the RDDS.

The IMS Repository requires the RM to be at a minimum level of IMS 12.

There is coexistence maintenance associated with this for IMS 12, which ensures that all RM address spaces are at IMS 12 for users of the IMS Repository.

For IMS 13, it is included in the base.

10.6.3 IMS Repository

If automation includes IMPORT with SCOPE(ALL), account for the fact that it now applies to inactive IMS systems.

IMS change list is created and applied at warm start or emergency restart

IMS 12 considerations

IMS 12 requires APAR PM80588 for coexistence with IMS 13. It is required by all IMS 12 users of the IMSRSC repository in a mixed environment of IMS 12 and IMS 13. It also must be applied before IMS 13 APARs PM77568 and PM88861 were applied.

APAR PM80588 is not required if there is no IMS 12 SCI, RM, or OM in the IMSplex

An IMS 12 RM address space should not support the QUERY, UPDATE, and DELETE of an IMS change list function from the IMSRSC repository because these functions are not supported in IMS 12.

APAR PM80588 adds logic to disable support and reject any QUERY, UPDATE, or DELETE of the IMS change list in the IMSRSC repository because these functions are not supported in IMS 12.

IMS 13 repository changelist enhancement

APARs PM77568 and PM88861 add IMS repository change list enablement support to all IMS 13 users of the IMSRSC repository.

The code for the IMSRSC repository change list function is in the IMS 13 base, but is disabled. These APARs add code to enable the IMS repository change list function and fix several issues that are found.

Support also is added for the DELETE DEFN TYPE(CHGLIST) command.

10.6.4 IMS user exits

Include exits that you intend to add, update, or delete in the IMS runtime environment in the USER_EXITS section of DFSDFxxx proclib member. You can use the enhanced services for some exit types while you continue to use non-enhanced services for other exit types.

If you are using exits that do not use the enhanced user exit services, you continue to use these exits.

Enhanced user exit services are optional for existing exits.

To activate the enhanced services for an exit type that is newly supported in IMS 13, complete the following tasks:

- Add the exits to DFSDFxxx proclib member.
- Issue a REFRESH USEREXIT command.

Enhanced user exit services are then available for use.

Migration is on a user exit type basis. A single user exit type can be migrated to use the new services while the other user exit types can continue with the non-enhanced services.

10.6.5 IMS SECURITY macro removal

The SECURITY macro is removed from IMS 13 SYSGEN. New parameters were added in DFSPBxxx and DFSDCxxx proclib members. For more information, see 2.3, “IMS SECURITY macro removal” on page 33.

User exits removed from IMS nucleus

Exits DFSCSGN0, DFSCTRN0, DFSCTSE0 are no longer in DFSVNUCx. Consider the following points:

- Exit DFSCSGN0 is not in the nucleus, it is stand-alone and it is loaded at IMS start.

  In IMS 13, we call DFSCSGN0 at IMS initialization. The exit can return an address of shared storage. That address is passed to DFSCSGN0, DFSCTRN0, and DFSCTSE0 for every call.
- If the exit routines cannot be linked separately or cannot use a common work area, they must be linked in the following manner:
  - If the CSECT of DFSCTSE0 is part of DFSCTRN0 source, DFSCTSE0 must be linked as an ALIAS of DFSCTRN0.
  - If virtual address spaces are used to exchange data between DFSCSGN0, DFSCTRN0, and DFSCTSE0, both DFSCTSE0 and DFSCSGN0 must be linked as an ALIAS of DFSCTRN0.

Apply SPEs for RCLASS/SECCNT support

To define RCLASS and SECCNT as execution parameters in DFSPBxxx/DFSDCxxx members, ensure that the following PTFs are applied:

- PM48203 (UK74050) and PM72199 (UK82616) for IMS 11
- PM48204 (UK74051) and PM73558 (UK82617) for IMS 12

Removing the SECURITY macro

Because the SECURITY macro is removed, all parameters that are defined there must be defined as execution parameters except for the exit-related parameters TYPE=TRANEXIT and TYPE=SIGNEXIT.

These exit-related parameters specify that an exit is called for transaction and sign-on authorization and they do not have start parameter equivalents.

To continue using transaction authorization and sign-on authorization, define the associated exits in STEPLIB and they automatically are called without requiring that a parameter exists that indicates that an exit is used.

If the exit-related parameters are left on the SECURITY macro in IMS 13, they are ignored.

You can remove the SECURITY macro entirely from STAGE1 before migrating to IMS 13 only if you are not using the exits.

10.6.6 DBCTL, ODBA, and ODBM migration

The Database Resource Adapter (DRA) interface modules are DFSPRRC0 and the DFSPZPxx member, which is the DRA start table that was created by assembling the DFSPRP macro and link-editing it into the IMS SDFSRESL library (or another APF-authorized library) as DFSPZPxx, where xx=00 is the default.

These modules are specific to a version of IMS and must be at the same version as IMS. They are in the address space that communicates with IMS. These address spaces can include CICS, ODBM, and the address spaces that uses ODBA, such as DB2 stored procedures and WebSphere Application Server.

Copy DFSPRRC0 and DFSPZPxx load modules to the address spaces which communicate with IMS. Include these modules from IMS 13 in the CICS, ODBM, and ODBA (DB2 Stored Procedures, WebSphere Application Server, and so on) address spaces.

10.6.7 RSR migration

You can migrate all of the Remote Site Recovery (RSR) components at the same time, but it is more likely that migrating them in stages is preferred. The restrictions that are described in 10.1.20, “RSR coexistence” on page 274 imply that the components must be migrated in the following order:

- The tracking system must be migrated before or at the same time as the ILS at the active site.
- The ILS at the active site must be migrated before or at the same time as the active IMS system.
- The RECONs must be upgraded to IMS 13 before the systems that use them are migrated to IMS 13.

Complete the following steps to migrate RSR components:

1. Upgrade the RSR tracking system RECONs to IMS 13.
2. Migrate RSR tracking system to IMS 13.
3. Upgrade the active system RECONs to IMS 13.
4. Migrate active system Transport Manager Subsystem (TMS) that is running Isolated Log Sender to IMS 13.

5. Migrate active IMS to IMS 13.

10.6.8 RACF password phrase support

User exits might need to be changed to handle password phrases.

If you are not using RACF, check with vendors to determine whether the External Security Manager supports password phrases.

This functionality is provided with IMS 12 SPE APAR PM91898 (TM RA V13.2.0).

> **Note:** IMS Connect V13 PM91312 and IMS V13 PM85849 are required for this function to work properly.
>
> Both APARs can be applied independently to TMRA and IMS Connect; however, both APARs must be applied to use the password phrase feature.

10.6.9 OTMA global flood control

Older releases of IMS provided the following messages for an input message flood condition:

- DFS1988W
- DFS1989E
- DFS0767I

In IMS 11, the following messages were introduced:

- DFS4380W
- DFS4381I
- DFS4388W
- DFS0793I

IMS 13 OTMA messages

In IMS 13, there are enhancements to DFS4388W and DFS0793I messages with possible changes to automation programs that manage DFS4388W and DFS0793I messages.

There are also two new messages: DFS3428W and DFS3429E.

In IMS 13, the existing DFS4388W (when limit is reached and the new global support is not active) and DFS0793I (when flood is relieved) messages were enhanced to make them clearer and to be consistent with the new messages, as shown in the following example:

```
DFS3428W THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED XX% OF THE GLOBAL LIMIT
ZZZZ
DFS3429E THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED THE GLOBAL LIMIT ZZZZ
```

/STA TMEMBER ALL INPUT command

In previous releases, the command /STA TMEMBER ALL INPUT #### resulted in warning messages. It did not trigger the flood status update in the /DIS OTMA; therefore, new input continued to be accepted.

With IMS 13, the flood status takes effect immediately. If there are more messages waiting to be processed than the value zzzz specified, the following events occur:

- Message DFS3429E is displayed (THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED THE GLOBAL LIMIT ZZZZ).

- /DIS OTMA shows a status of SERVER+FLOOD.
- New input is rejected.

> **Important:** There is immediate effect if you change the value to be less than what is queued.

10.6.10 IMS Connect

Changes were made for OTMA TCO, XML converters, and record trace.

IMS Connect/OTMA TCO impact

The conversion of OTMA and IMS Connect STORAGE calls to CPOOL in its XCF SRB exits improves performance by being less CPU-intensive and faster in memory acquisition.

The change adds 167 MB of virtual storage in EVPT for IMS Connect and 256 KB for IMS control region.

XML converters increased

The previous maximum of 100 XML converters that can be loaded at any one time into an IMS Connect instance was increased from 100 to 2000.

To support the new capability, the existing IMS Connect command output to display the configuration was enhanced to include the following new fields:

- Current number of converters
- Maximum limit

This addition affects the output of the following commands:

- The **VIEWHWS** WTOR command
- The z/OS Modify command **F HWS1,QUERY MEMBER TYPE(IMSCON) SHOW(ALL)**

  Figure 10-23 shows the output of the (WTOR) **VIEWHWS** and z/OS MODIFY **QUERY** commands.

```
HWSC0001I   HWS ID=HWS1      RACF=N  PSWDMC=R
HWSC0001I      UIDCACHE=N    UIDAGE=2147483647
HWSC0001I      MAXSOC=50      TIMEOUT=2147483647
HWSC0001I      NUMSOC=5       WARNSOC=55%  WARNINC=5%
HWSC0001I      RRS=Y   STATUS=REGISTERED
HWSC0001I      VERSION=V13 IP-ADDRESS=009.030.114.199
HWSC0001I      SUPER MEMBER NAME=      CM0 ACK TOQ=
HWSC0001I    ADAPTER=Y MAXCVRT=400  NUMCVRT=15
```

Figure 10-23 Output of VIEWHWS and z/OS MODIFY commands

- The IMS type-2 command **QUERY IMSCON TYPE(CONFIG) SHOW(ALL)**

  Figure 10-24 on page 315 shows the output of QUERY type-2 command.

```
QUERY IMSCON TYPE(CONFIG) SHOW(ALL)

TSO SPOC output:  (screen 1)
 MbrName CC  Version IconID IpAddress                MaxSoc TimeOut NumSoc   WarnSoc   WarnInc
 HWS1       0       V13      HWS1   009.030.124.150     50         5000         4               80
5

....
(screen 3)
 MbrName Adapter MaxCvrt NumCvrt ODBMAC ODBMTO ODBMIpMem ODBMTMem
 HWS1        Y           400      15                    Y            18000        IMSPLEX1
PLEX1
```

Figure 10-24   Output of QUERY type-2 command

Recorder trace expanded

IMS Connect 12 introduced new Recorder Trace records that used BPEs external trace tables.

In IMS Connect 13, these trace records were expanded to capture the entire messages that are sent and received for all TCP/IP communications that include DRDA sockets and ISC TCP/IP and SCI (OM, ODBM, MSC, and ISC communications).

Enabling the trace in IMS Connect 13 uses the same command as in IMS Connect 12; however, the amount of trace information that is produced can be significantly higher, as shown in the following example:

```
UPDATE TRACETABLE NAME(RCTR) OWNER(HWS) LEVEL(HIGH) EXTERNAL(YES | NO
```

Ensure that BPE external tracing is enabled to account for the increase.

10.6.11 ISC TCP/IP migration and fallback considerations

The steps for migrating ISC terminals to use TCP/IP differ depending on whether the terminals are statically or dynamically defined.

Static ISC terminals

In this section, we examine migration and fallback for static ISC terminals.

Migration

Complete the following steps to migrate statically defined ISC terminals to TCP/IP:

1. Ensure that the IMS definitions are correct.

   If you are switching an existing terminal from VTAM to TCP/IP, ensure that the terminal is defined with UNITTYPE=LUTYPE6 in the TYPE stage-1 system definition macro.

   If you are defining a new terminal, code the appropriate system definition macros, including the TYPE, TERMINAL, and SUBPOOL macros. The TYPE macro must specify UNITYPE=LUTYPE6.

2. Specify ISCTCPIP=(nodename,iconname) in the DFSDCxxx PROCLIB member for each node that is to use the TCP/IP support.

> **Note:** Existing VTAM connection definitions in CICS must be changed to use IPIC if the same names are being used.

3. Cold start IMS.

Fallback

Complete the following steps to revert statically defined terminals to VTAM support:

1. Delete the ISCTCPIP parameter that specifies the node name of the terminal from the DFSDCxxx PROCLIB member.
2. Cold start IMS. Warm and emergency restarts keep the ISCTCPIP specification active.
3. Review the CICS requirements.

For dynamic ISC terminals

In this section, we examine migration and fallback for static ISC terminals.

Migration

Complete the following steps to migrate dynamically and ETO defined ISC terminals to TCP/IP:

1. Define ETO logon descriptors that specify UNITYPE=ISCTCPIP and LCLICON=ims_connect.

   Create a generic descriptor that can be used by several nodes.
2. Make the associated definition changes in CICS.
3. Run the following IMS type-1 command:

   ```
   /OPNDST NODE node1 USER user1 LOGOND logondescriptor
   ```

Fallback

Complete the following steps to revert dynamically defined terminals to VTAM support:

1. Stop the TCP/IP ISC connections by issuing either IMS type-1 command **/STOP** or **/QUIESCE**.
2. If necessary, put the USER (subpool) that is associated with the TCP/IP connection into a cold state by issuing the IMS type-1 command **/ASSIGN USER *user1* TO VTAMPOOL**.

   The command deallocates the USER user1 from the TCP/IP connection so that it can be used for the logon of ISC VTAM connections.

   > **Note:** The USER cannot be signed on or in conversation mode, response mode, or preset mode.

3. Restart the ISC VTAM connection by running the following IMS type-1 command:

   ```
   /OPNDST NODE node1 USER user1
   ```

10.6.12 IMS Enterprise Suite

In this section, we summarize the changes for Enterprise Suite for the various platforms.

IMS Enterprise Suite V2.2 for z/OS

A new IMS Enterprise Suite 2.2 for use with z/OS V01.12.00 or higher is now available. The components in IMS Enterprise Suite for z/OS V2.02.00 are available for the z/OS or distributed platforms to facilitate application development and enable greater reach and maximum participation of IMS transactions and data in an on-demand or service-oriented business environment.

The Enterprise Suite for z/OS is supplied as no-charge orderable Custom-Built Product Delivery Offering (CBPDO) with program number 5655-T62. For more information, see the *Program Directory for IMS Enterprise Suite*, GI10-8905-01.

IMS Enterprise Suite for z/OS consists of the following FMIDs:

- HAHF220 (Base Services)
- JAHF221 (SOAP Gateway)
- JAHF222 (Java Message Services API)
- JAHF223 (Connect API Java)

Components

The following components run on z/OS that is supplied in CBPDO:

- Base Services includes sample jobs to use for the installation of z/OS based components and the IBM 31-bits software development kit (SDK) for z/OS, Java Technology Edition Version 7. The IBM SDK is a set of software development tools with which Java applications can be created.
- SOAP Gateway enables IMS applications to interoperate outside of the IMS environment through the SOAP protocol to provide and request services.
- Connect API provide programming control of connections to IMS Connect.
- Java Message Services (JMS) API is included for an IMS application that runs in the Java message processing (JMP) or Java batch processing (JBP) region to issue synchronous callout requests to external services.

IBM Installation Manager

IBM Installation Manager helps you install, update, modify, roll back, and uninstall packages on your computer. IBM Installation Manager for z/OS is UNIX System Services application.

Before IBM Installation Manager is used, you should customize your z/OS system as described in *z/OS UNIX System Services Planning* manual, GA22-7800.

The IBM Installation Manager V01.05.03 or higher is required to install SOAP Gateway. CBPDO ships with V01.04.03. Apply APAR PM66543 (PTF UK79476) to IBM Installation Manager V01.04.03 to upgrade it to V01.05.03

IMS Enterprise Suite V2.2 for distributed platforms

The IMS Enterprise Suite for Distributed Systems V2.02.00 has the following components that are available as downloads:

- Connect API for C
- Connect API for Java
- IMS Explorer for Development

  Enterprise Suite Explorer for Development simplifies IMS application development tasks by displaying and enabling editing of IMS databases, segments, fields, and more from an industry-standard integrated development environment (IDE).
- SOAP Gateway

  Components for the distributed platform are installed by using the standard IBM Installation Manager. For more information, see this website:

  `http://www.ibm.com/ims`

IMS Enterprise Suite 2.3 .NET

The data provider system requirements are described in this section.

Hardware requirements

For the IMS 13 host system, see IMS 13 release planning information.

The IMS .NET Data Provider and Microsoft Visual Studio includes the following requirements:

- 1.6 GHz or faster processor
- 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512 MB if running in a virtual machine)
- 3 GB of available hard disk space

Software requirements

The following software is required:

- IMS 13:
  - IMS Connect
  - CSL, Open Database Manager (ODBM)
  - IMS catalog
  - IMS Enterprise Suite 2.3 .NET Data Provider
- Microsoft Visual Studio 2010
- Windows XP with Service Pack 3 or later (32 or 64 bit)

Distribution site

For more information, see this website:

http://www-01.ibm.com/software/data/ims/soa-integration-suite/enterprise-suite

IMS .Net Data Provider installation

Use the IBM Installation Manager to install the IMS .NET Data Provider. The IMS .NET Data Provider is distributed as a dynamic link library (`ImsDataProvider.dll`).

Add the `.dll` to the following client application reference path:

```
C:\Program Files\ReferenceAssemblies\Microsoft\Framework\.NETFramework\v4.0
```

IMS Enterprise 2.2 Connect API for Java

The following software versions are supported:

- z/OS V1.12 or later
- IMS 13 ,12 and 11
- Microsoft Windows XP (SP2)

IMS Enterprise Suite 2.2 SOAP Gateway

The following IMS versions are supported:

- IMS Enterprise Suite Version 2.2 supports IMS 13, 12 ,11
- For IMS 11, APAR PM27324 must be applied
- For IMS 12, APAR PM29137 must be applied

  The SOAP Gateway transaction tracking and transaction logging features requires IMS 12 APAR PM69983

Rational Developer for System z Version 8.0.3.2 or later is required.

The business events include the following requirements:

- For WebSphere Business Events:
  - WebSphere Business Events Development Toolkit Version 6.2 or later that is installed in a compatible version of IBM Rational Application Developer for WebSphere Software
  - WebSphere Business Events Server Version 6.2 or later
- For Business Monitor

  Business Monitor Development Toolkit Version 6.2 or later that is installed in a compatible version of Rational Application Developer

IMS Enterprise Suite 2.2 SOAP Gateway

The correlator schema changed in IMS Enterprise Suite Version 2.2 SOAP Gateway. For information about migration and maintenance, see IMS Enterprise Suite Version 2.2 SOAP Gateway README, which is available at this website:

http://www-01.ibm.com/support/docview.wss?uid=swg27027299

Migrating existing web services

When you upgrade to IMS Enterprise Suite Version 2.2, the process of migrating existing web services by using the SOAP Gateway management utility **`iogmgmt -migrate path_to_source_installation`** command handles the correlator migration.

Migrating existing web services

For newly generated web service artifacts, you must run the **`iogmgmt -migrate`** correlator command to migrate the correlator files to the new schema before deploying the web services. The **`iogmgmt -migrate`** now supports the migration of server properties.

Enhanced migration support of server properties

To migrate from version 2.1, you specify the absolute path to the installation of IMS Enterprise Suite Version 2.1.

10.6.13 Database versioning

Before database versioning is enabled for a database, IMS continues to recognize only the current active database definition. To activate versioning, you must populate the catalog with DBD version definitions

In preparation to enable database versioning, specify the following new DBD and PSB statement versioning attributes:

- Use the DBLEVEL=CURR setting so all applications access database at latest levels by default.
- Use the PCB DBVER= parameter if an application needs to continue using an old version of a database.
- All systems in an IMSplex must be running IMS 13.

10.6.14 HALDB alter

Migration includes the following considerations:

- All systems in an IMSplex must be running IMS 13.

- Enable Common Service Layer (CSL):
  - Operations Manager (OM)
  - Structures Call Interface (SCI)
  - Type-2 INITIATE OLREORG command with new OPTION(ALTER)
  - Type-2 QUERY OLREORG command to display ALTER status for a DB
- DBRC:
  - RECON MINVERS must be at least ('13.1')
  - Several DBRC LISTs produce output that contains new fields
  - DBRC API Query output changed
  - Content of several DBRC RECON records changed
- If database ALTER is needed and HALDB is not used, convert Full-Function database to HALDB database.

10.6.15 AIB INQY ENVIRON call

The Inquiry (INQY) call is used to request information regarding execution environment, destination type and status, and session status.

The IMS 12 APAR PM42909 (PTF UK78069) expanded the response area for AIB INQY ENVIRON call from a minimum of 100 bytes to 108 bytes.

Application programs that are running in IMS 12 and above might need to be changed to specify a larger I/O area and recompiled.

Application programs that issue the INQY ENVIRON call that have an area less than the minimum size receive a status code "AG".

Consider the following points:

- The I/O area is filled with partial data, as much as can fit in the I/O area that is provided.
- AIBOALEN contains the length of the data that is returned in the I/O area.
- AIBOAUSE contains the output area length that is required to receive all of the data for the INQY ENVIRON call.

To receive the reply data and to account for expansion, define the I/O area length to be larger than 160 bytes. If you define the I/O area length to be exactly 160 bytes and the I/O area is expanded in future releases, you receive an AG status code.

> **Note:** Recovery Token and APARM are optional fields that, if present, follow ENVIRON data. Do *not* use offset to get to Recovery Token or APARM; instead, use pointers that are contained within the ENVIRON data. Otherwise, the application must be adjusted as needed to accommodate offset changes.

10.6.16 FPBP64=Y Fast Path 64-bit buffer manager

In IMS 11, you can enable the Fast Path 64-bit buffer manager by specifying FPBP64=Y in the Fast Path section of DFSDFxxx PROCLIB member, as shown in Example 10-3.

Example 10-3 IMS 11 Fast Path specifications in DFSDFXXX

```
<SECTION=FASTPATH>
FPBP64=Y,FPBP64M=xxxxxx (set maximum storage used)
```

When the Fast Path 64-bit buffer manager is enabled, IMS ignores the DBBF, DBFX, and BSIZ parameters, if specified. If FPBP64=Y is specified, initially turn off 64-BIT DEDB buffer compression and pre-expansion.

Complete the initial migration without implementing this optional new function. After it is stable in production, use the new function.

Consider the following points:

- IMS 12 APAR PM84873 (PTF UK92995) changes the default subpool compression and pre-expansion setting for the FP 64-bit buffer manager from ON to OFF (default to ‘N’).
- If PM84873 is applied, specify FPBP64C=Y and FPBP64E=Y in the DFSDFxxx member to enable compression and pre-expansion.

11

IMS Enterprise Suite

The IMS Enterprise Suite is a set of components that support open integration technologies to enable new application development and extend access to IMS transactions and data.

This chapter includes the following topics:

- IMS Enterprise Suite components
- Role of the IMS Enterprise Suite Explorer for Development
- Using IMS Explorer to capture IMS metadata
- Using IMS Explorer to Access IMS Data
- IMS Enterprise Suite SOAP Gateway
- IMS Data Provider for Microsoft .NET

11.1 IMS Enterprise Suite components

The IMS Enterprise Suite Version 3.1 includes the following components:

- IMS Mobile Feature Pack

  IMS Mobile provides the solution to easily enable your IMS transaction assets as services for mobile and cloud use.

- IMS Explorer for Development

  Simplifies IMS application development tasks, such as updating IMS database and program definitions, and uses standard Structured Query Language (SQL) to manipulate IMS data in an Eclipse-based graphical environment.

- IMS Data Provider for Microsoft .NET

  Supports the use of standard SQL queries to access IMS data from .NET applications.

- SOAP Gateway

  Enables IMS applications to interoperate outside of the IMS environment through the SOAP protocol to provide and request services that are independent of platform, environment, application language, or programming model. IMS applications can become web service providers or users in a service-oriented business environment. SOAP Gateway also enables IMS applications to emit business events data to business event processing engines.

- Connect API for C

  Application programming interfaces for custom IMS Connect TCP/IP client applications that are written in C or C++ to specify and configure connections and interactions with IMS Connect.

  There are no changes to the Connect API for C with IMS Enterprise Suite Version 3.1.

- Connect API for Java

  Application programming interfaces for custom IMS Connect TCP/IP client applications that are written in Java to specify and configure connections and interactions with IMS Connect.

  There are no changes to the Connect API for Java with IMS Enterprise Suite Version 3.1.

- Java Message Service (JMS) API

  Issues synchronous callout requests to external services from within a Java message processing (JMP) or Java batch processing (JBP) application that runs in IMS Version 11 or later.

  There are no changes to the Java Message Service (JMS) API with IMS Enterprise Suite Version 3.1.

IMS Explorer for Administration is an extension to the IBM Tools Base Administration Console for System z. It provides tools to gather information about and administer IMS transactions, databases, application programs, Fast Path routing codes, and IMSplex members. You can use the IMS Explorer for Administration to query and start and stop IMS resources from a browser-based interface. The IMS Explorer for Administration does not require any other IBM IMS Tools products.

11.2 Role of the IMS Enterprise Suite Explorer for Development

The IMS Enterprise Suite Explorer for Development is an Eclipse-based graphical tool that simplifies many IMS application development tasks, such as updating IMS database and program definitions. The tool uses standard SQL to manipulate the IMS data.

You can use the IMS Explorer graphical editors to import, visualize, and edit IMS database and program definitions. You can also use the tool to easily access and manipulate data that is stored in IMS by using standard SQL.

The IMS Explorer can also directly import DBDs and PSBs, or can obtain existing catalog information from the IMS catalog through a type-4 connection.

The population of the IMS catalog is done from the PSB and DBD sources. Other **DBDGEN** statements and attributes on the FIELD macro can provide complete metadata information.

Other metadata can be collected from COBOL copybooks and PL/I or C include members. After the metadata information is updated in the PSBs and DBDs in the IMS Explorer, those sources can be sent to the host for the GEN process. This process populates the metadata into the IMS catalog.

Figure 11-1 shows how the IMS Explorer can interact with the IMS catalog directly through a type-4 driver connection. Figure 11-1also shows an interaction with the catalog indirectly through a File Transfer Protocol (FTP) import and export.

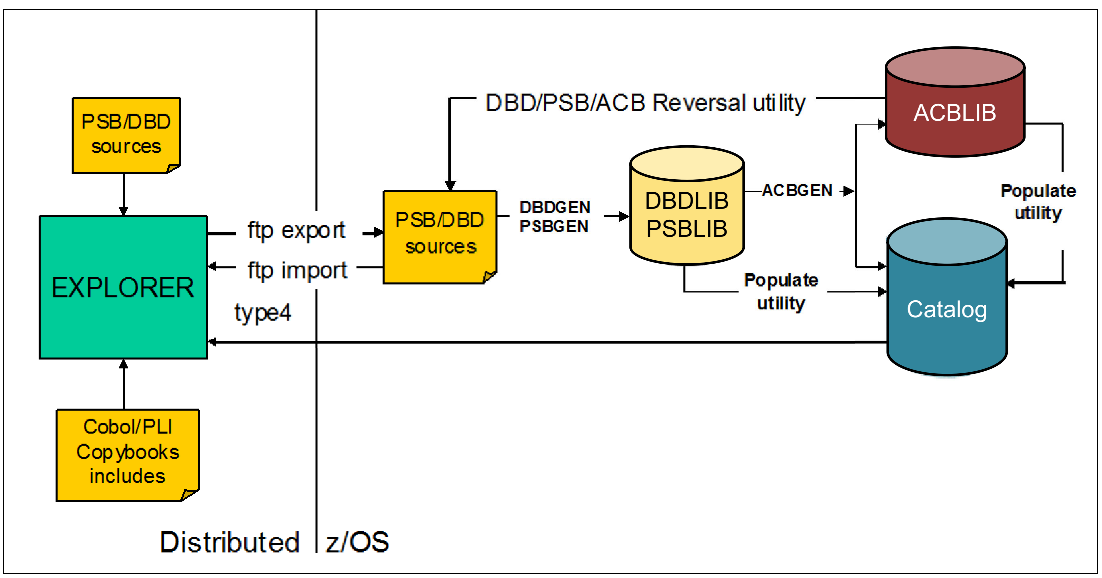


Figure 11-1 IMS Explorer and catalog

A direct update from the IMS Explorer is not available. An intermediate application control block generation (ACBGEN) with information from the catalog is required.

The DLIModel utility is a helpful tool for visualizing the structure of an IMS database. The tool documents the details of the DBD and PSB sources. The DLIModel utility is required to produce the `com.ibm.ims.db.DLIDatabaseView` class for accessing the IMS databases from Java applications on the remote and local locations. This access is through the type-4 and type-2 drivers. The traditional DL/I SSA and JDBC access can be used.

The DLIModel utility was not enhanced. All of the functions of DLIModel were integrated into the IMS Enterprise Suite Explorer for Development. The IMS Explorer uses the centralized IMS catalog on IBM z/OS. The metadata that is available in the DatabaseView class is integrated in the IMS catalog. The concept of the IMS catalog makes the metadata more dynamic and shared. Its implementation avoids the need for distributing the class to all sites where it might be used in Java programs with DL/I (or JDBC) access. The access is through the IMS database type-2 and type-4 Universal drivers. The drivers were adapted to provide this function.

The following features summarize the IBM Enterprise Explorer:

- Incorporates DLIModel functionality.

  Support for migration of DLIModel projects is provided.
- Provides the following graphical editors for development and visualization:
  - Database Description (DBD)
  - Program Specification Block (PSB)
- Shows a relational view of IMS data with graphical assistance to build SQL statements.

The IMS Universal JDBC type-4 connectivity extracts DBD and PSB information for local enhancement.

Input for the IMS Explorer

For more information about how the input for the IMS Explorer is taken from local PSB and DBD sources, see 11.2, “Role of the IMS Enterprise Suite Explorer for Development” on page 325.

The IMS Explorer can also take input from the following sources:

- Import of the DLIModel utility project
- Remote import
- The IMS catalog

All information about the PSBs and DBDs is consolidated into the IMS catalog. By using the new type-4 driver, the IMS Explorer can extract PSB and DBD information from the catalog to be updated locally. After the update, this information is sent back to the consolidating host by FTP. The Export window is shown in Figure 11-2 on page 327.

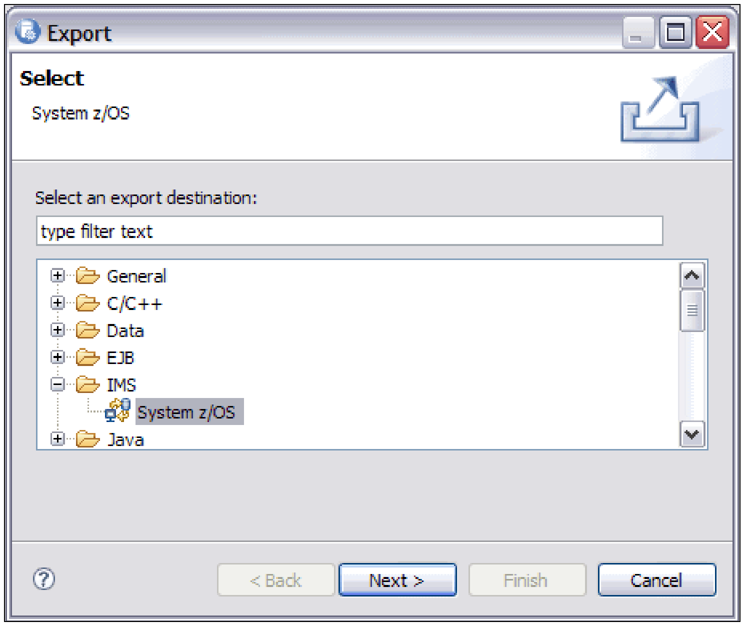


Figure 11-2 Export to the z/OS host with FTP

An ACBGEN of the changed PSB or DBD is needed to update the ACBLIB and populate the catalog with this updated information.

IMS Enterprise Suite Explorer and the IMS catalog

The IMS catalog is a PHIDAM database that contains trusted metadata for IMS databases and applications. All of the information that is contained in the ACBLIB run time (from the DBDLIB and PSBLIB) is available to users in the IMS catalog.

The IMS Explorer is an important tool for using and updating the information in the catalog. The IMS Explorer can access information from the catalog database, work with it, update it, and send it back to the IMS on z/OS for consolidation in the catalog.

The following data connections can be used between the IMS Explorer and catalog information:

- From z/OS:
  - FTP import from the DBD and PSB sources
  - Through type-4 driver direct access to the catalog
- To z/OS: FTP export to the DBD and PSB sources

11.2.1 Extending IMS database definitions with the IMS Explorer

This section describes the extension of an IMS DBD with the IMS Explorer for Development. By using the IMS Explorer, you can quickly and easily add metadata to the IMS databases. You can do this from the COBOL copybooks or PL/I include members that are used by application programs.

11.3 Using IMS Explorer to capture IMS metadata

The IMS Explorer facilitates the capture of more metadata for an IMS database than was previously stored in the IMS DBD. With IMS Explorer, the layout of the data in the database segments can be captured and added to the DBD. The enhanced DBD can then be generated back on the host system and included in the catalog.

This section describes how to use the IMS Explorer for Development to capture the extra metadata information from the COBOL copybooks. This section also describes how the enhanced DBD is ready for generation on the host system.

Before you start, you must install the IMS Explorer for Development. The code is available as part of the IMS Enterprise Suite Version 3, which is available at this website:

http://www.ibm.com/ims

There are several runtime versions of this code. The code that is needed for this task is the *shell-sharing* version. This version installs with the IBM Installation Manager as an extension to IBM Rational® Developer for zSeries® or IBM Rational Developer for zEnterprise.

Complete the following steps:

1. Run IBM Explorer for z/OS and open the IMS Explorer perspective, as shown in Figure 11-3.

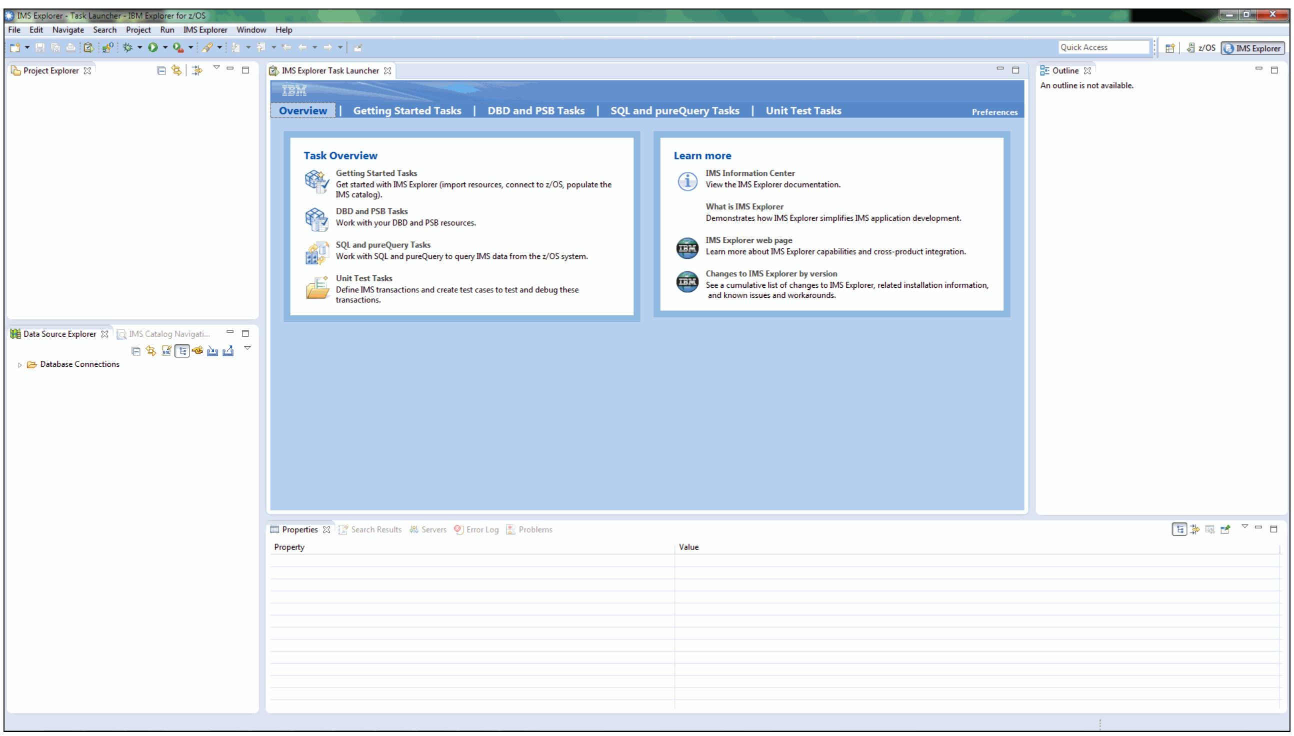


Figure 11-3 IMS Explorer perspective in Rational Developer for zEnterprise

2. To create an IMS Explorer Project, right-click in the Project Explorer window (upper left), and select **New** → **Other...** to start the Create a New Project wizard, as shown in Figure 11-4.

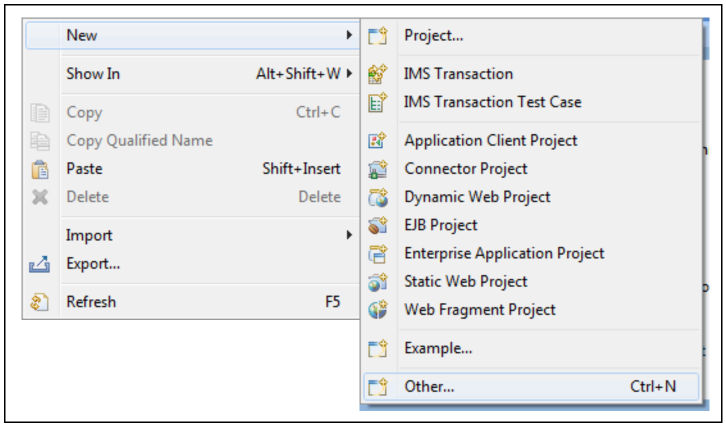


Figure 11-4 Create a New Project wizard

3. The wizard to be used is the IMS Explorer Project wizard. Select this wizard and click **Next**, as shown in Figure 11-5.

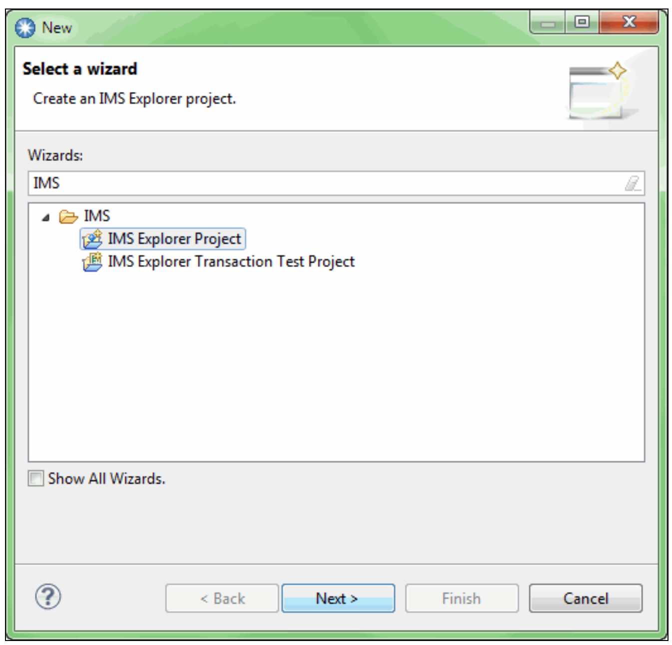


Figure 11-5 IMS Explorer Project

4. Enter a name for the IMS Explorer Project, as shown in Figure 11-6.

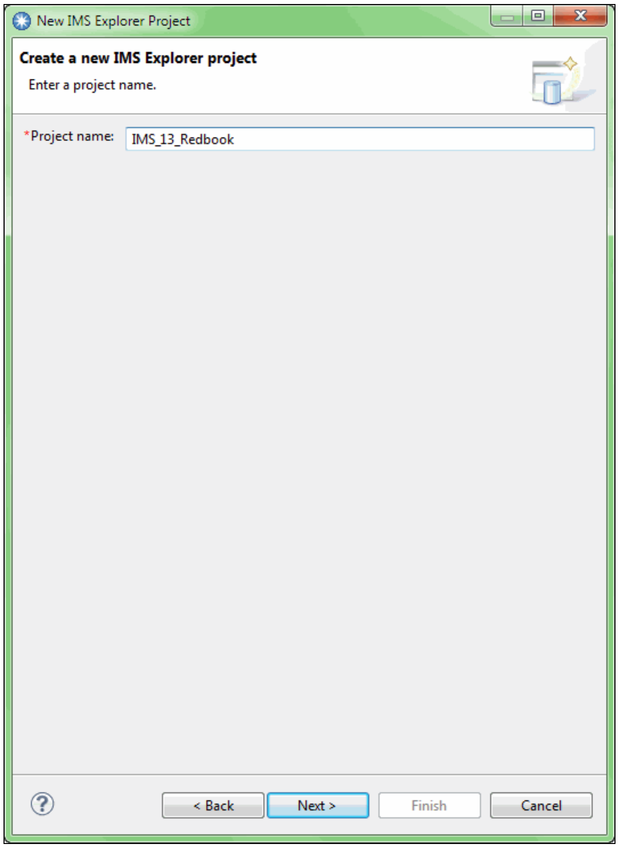


Figure 11-6 Name the IMS Explorer Project

When the wizard completes, IMS Explorer for Development prompts you for an input source for the named project. You then import the DBDs for the databases for which you want to capture the metadata.

When the wizard completes, IMS Explorer for Development asks for an input source for the To import the DBD option. Complete the following steps:

1. Continue with the Import Wizard, which prompts you for an import source, as shown in Figure 11-7.

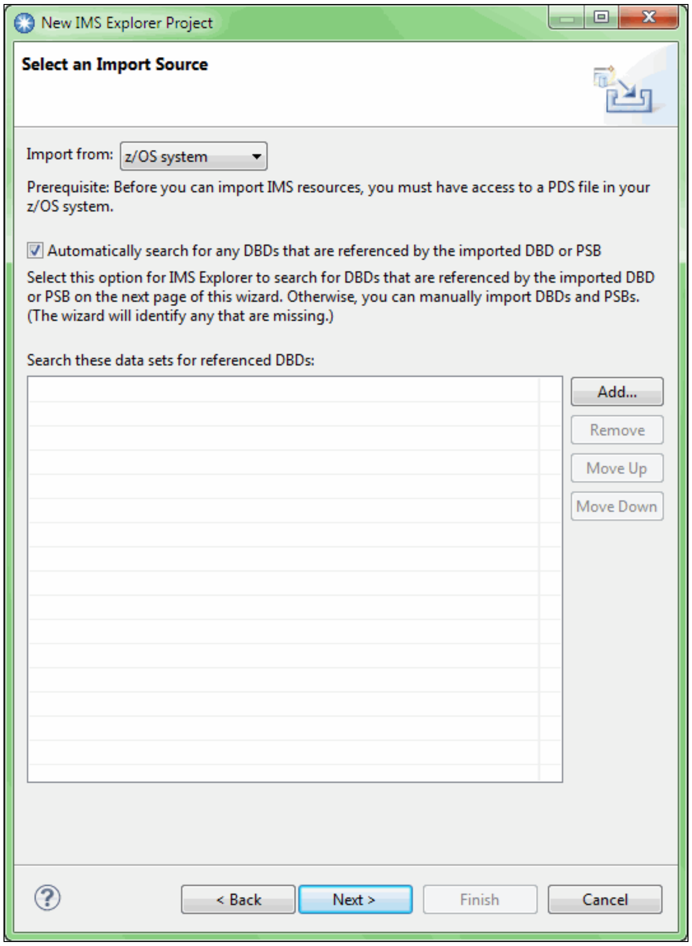


Figure 11-7 Select an import source for an IMS Explorer Project

2. The source for the DBDs and PSBs that you must import are on a z/OS system. If you downloaded them to your workstation or they are available on a LAN drive, you can select them from there. In the example that is shown in Figure 11-8, we set up an FTP connection to a z/OS host to download the DBDs to import.

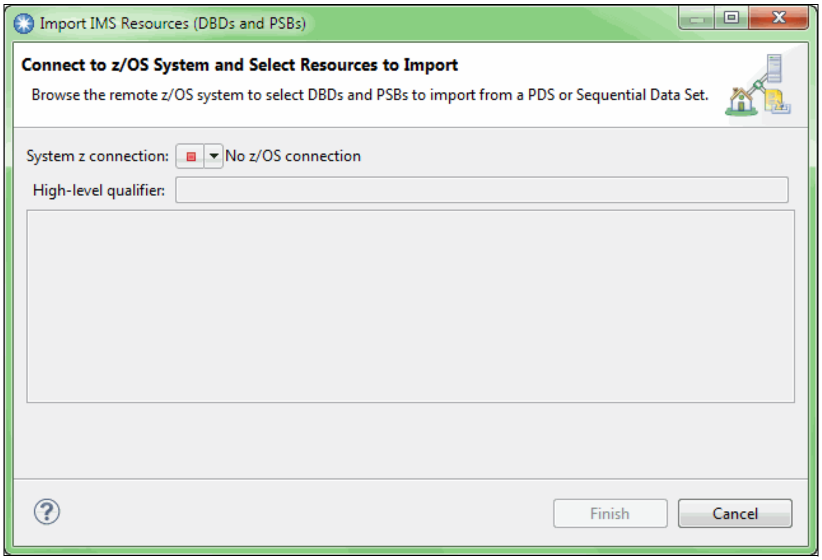


Figure 11-8 Connect to z/IOS System and Select Resources to Import

3. Select System z connection to choose a new (or existing) connection, as shown in Figure 11-9.

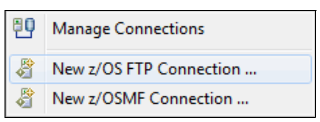


Figure 11-9 Select the connection to use

4. In the Add z/OS FTP Connection window (as shown in Figure 11-10), enter the details of your z/OS connection and select **Save and Connect**.

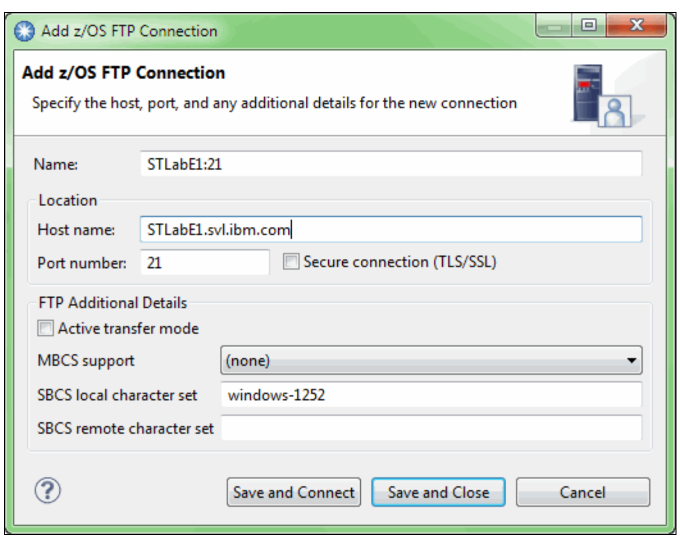


Figure 11-10 Add z/OS FTOP Connection

5. Enter your sign-on credentials, as shown in Figure 11-11. Click **OK**.

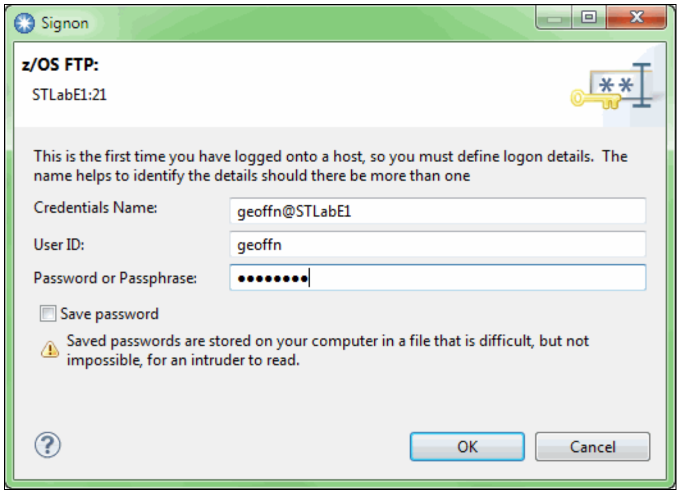


Figure 11-11 Sign on to z/OS FTP

6. As shown in Figure 11-12, click **Next** after you connect.

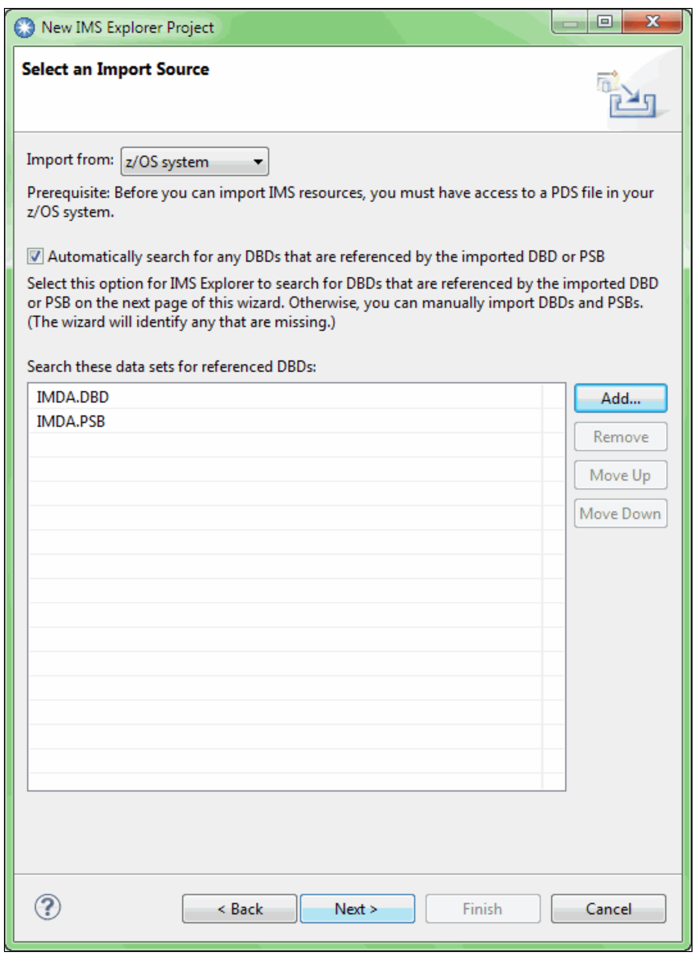


Figure 11-12 Select an Import Source

7. You can now select the files from the source that you want imported into your project. Although there are separate sections in this window for DBDs and PSBs, IMS Explorer can identify the resources in the source file that is selected, and then shows them in the appropriate list in this window. IMS Explorer also understands a file that contains JCL and multiple DBDs, PSBs, or a mixture of both, which simplifies the import process. Now that we are connected to our z/OS system, we can add the DBDs and PSBs that we need to import. Click **Add DBD**, as shown in Figure 11-13.

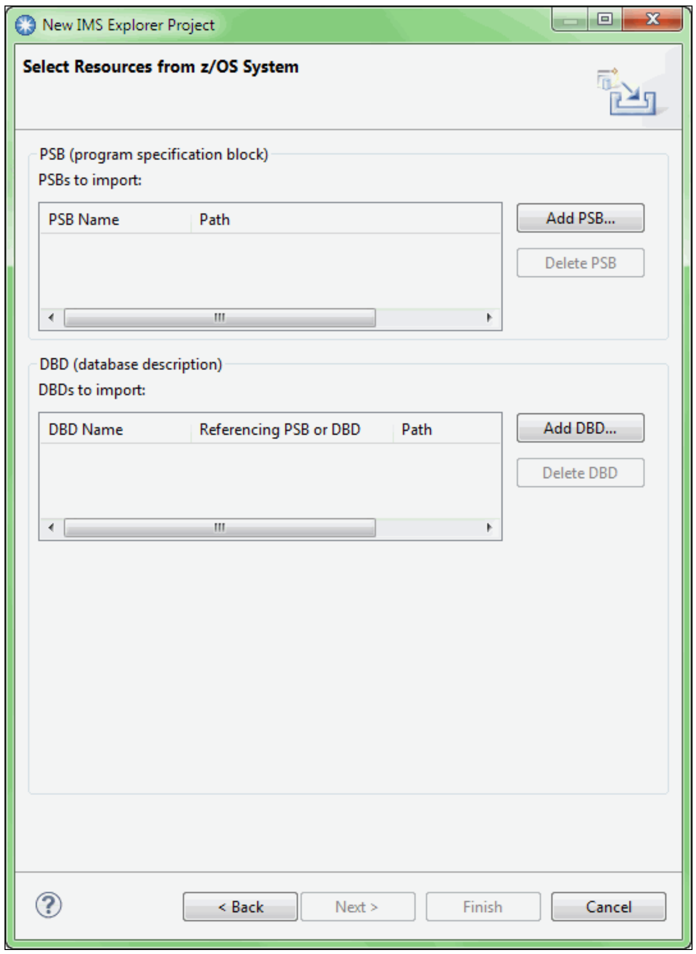


Figure 11-13 Select Resources from z/OS System

8. Browse to your DBD source library (as shown in Figure 11-14) and expand the list of members of this data set.

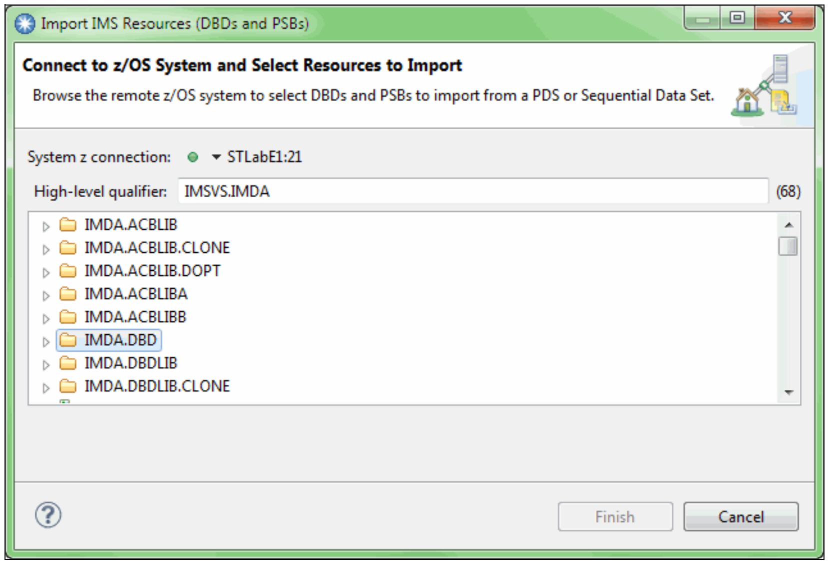


Figure 11-14 Import IMS Resources (DBDs and PSBs): Choose the DBD data set

9. Select the member for the database DBD that you want to import, as shown in Figure 11-15. Click **Finish**.

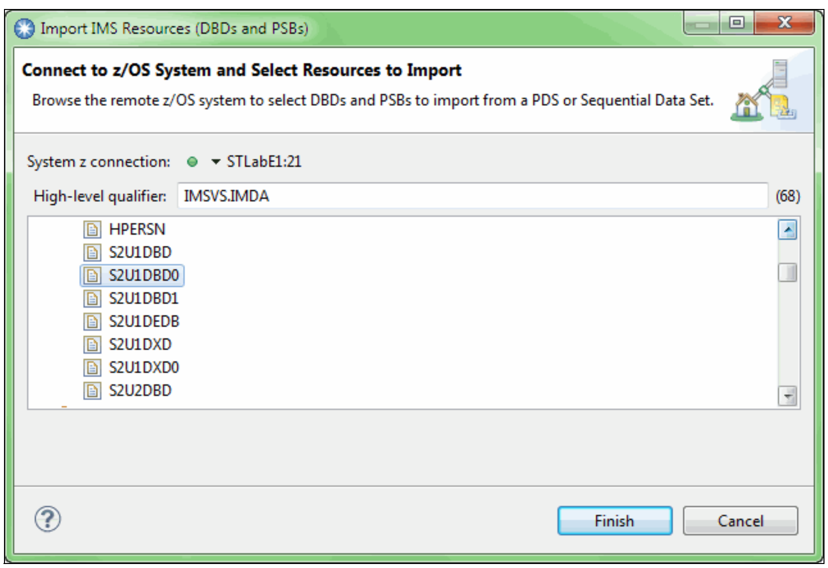


Figure 11-15 Import IMS Resources (DBDs and PSBs): Select the member

10. The Select Resources from z/OS System window shows the DBD that you selected and any related DBDs from the same library that are needed when it is imported into the project. Figure 11-16 on page 336 shows the S2U1DBD database and the associated Index DBD (S2U1DXD) because both of these DBDs are necessary for the S2U1DBD to generate correctly.

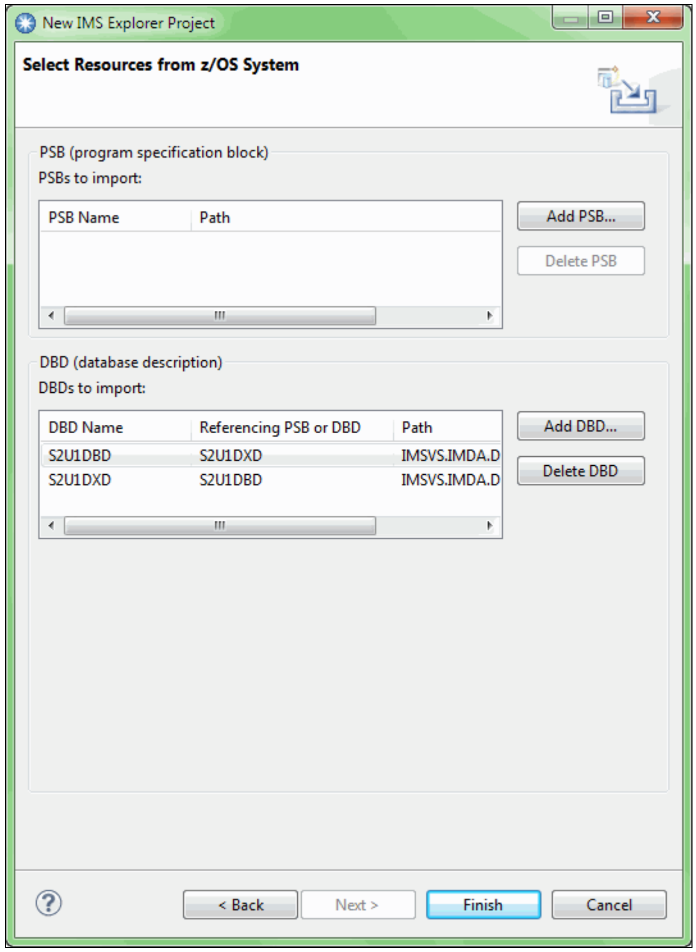


Figure 11-16 Select Resources from z/OS System: DBDs are selected

11. Use the same process to select the PSB (or PSBs) to be imported, as shown in Figure 11-17 on page 337. Any of these DBDs and PSBs can now be removed from the import list by clicking **Delete DBD** or **Delete PSB**.

    The IMS Explorer also shows any DBDs that still must be imported to satisfy the databases that are referenced in the PSBs to be imported. In this example, the two PSBs reference only the databases to be imported.

    Click **Finish** after you select all of the DBDs and PSBs that are to be imported.

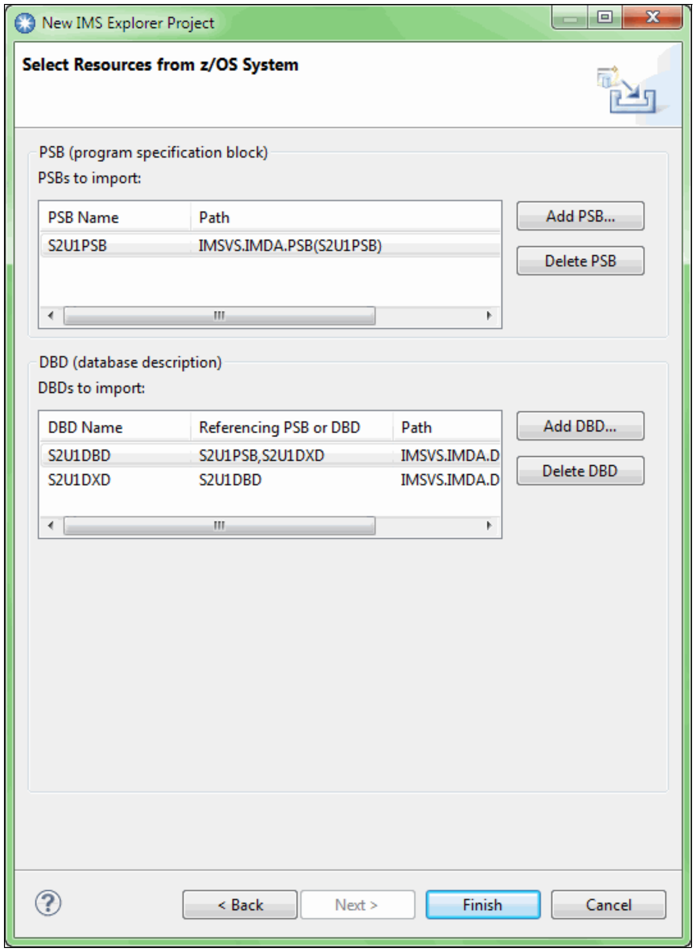


Figure 11-17 Select Resources from z/OS System: DBDs and PSBs are selected

12. You might receive a message box that informs you that there are segments missing in user-defined fields, as shown in Figure 11-18 on page 338. These fields are added to the DBD later. Click **OK**.

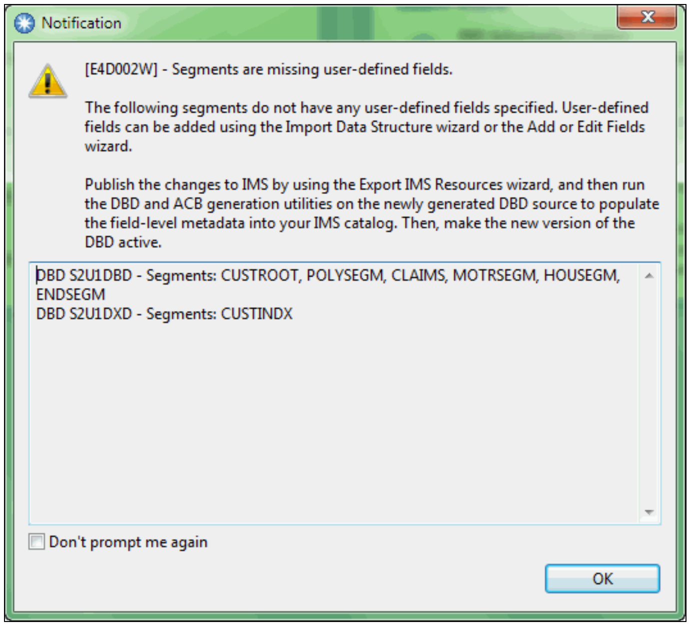


Figure 11-18 Notification

13. The project now includes the DBDs and PSBs that you imported and several other artifacts that are needed later. At any time, you can refer to this project to see the DBD or PSB source that was imported. The project also includes the generated source for the DBDs and PSBs. However, the generated source is the same as the imported source. The sources are the same because you did not yet add any of the metadata from the COBOL copybooks or PL/I include members.

    In the Project Explorer panel in the upper left of your workspace, expand the project name, and then expand the DBD and PSB to see the DBDs and PSBs that were imported into your project, as shown in Figure 11-19.

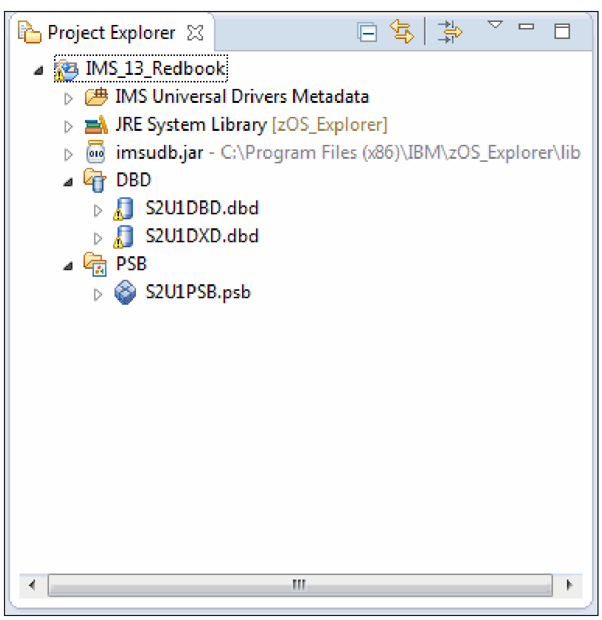


Figure 11-19 IMS Project Explorer, showing the DBDs and PSBs

14.To view a DBD in IMS Explorer for Development, double-click the DBD entry in your IMS Explorer Project. The graphical display is shown in Figure 11-20.

The DBD often does not include have many defined fields, apart from any required key fields.

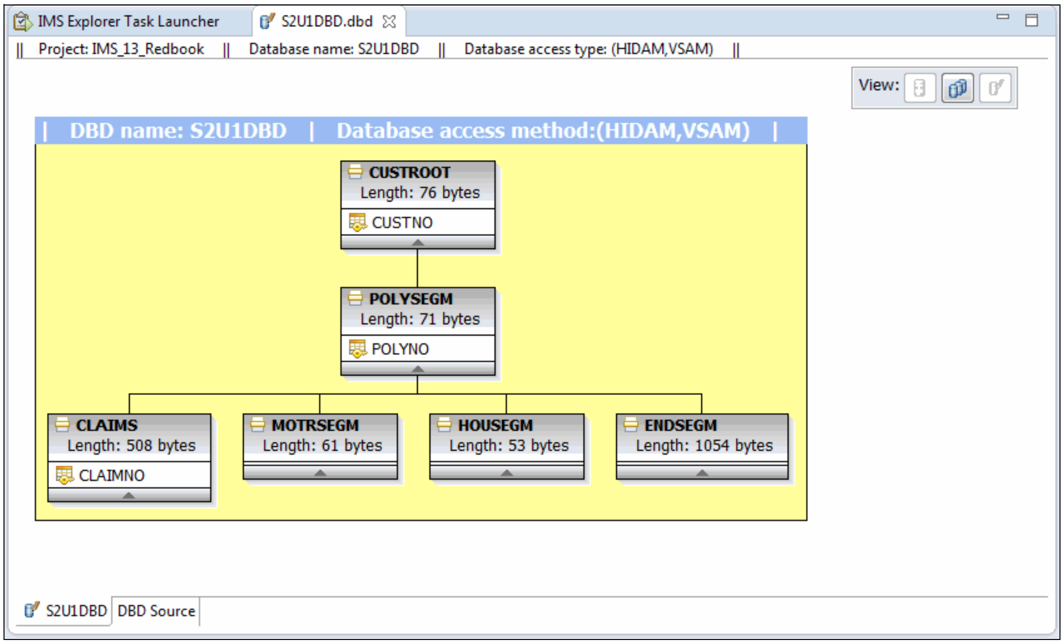


Figure 11-20 Graphical View of Database Layout

15.Click the **DBD Source** tab at the bottom to see the DBD statements for this database, as shown in Figure 11-21.

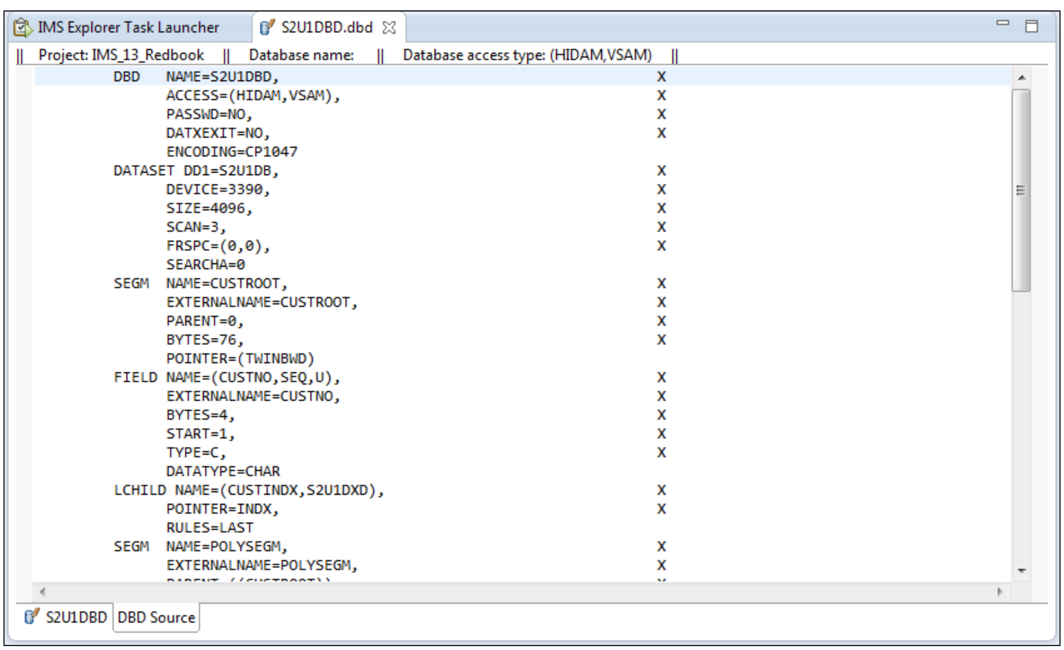


Figure 11-21 Source for the imported DBD

We now import the field metadata from the COBOL Copybooks or PL/I Include members that are used by your applications. Complete the following steps:

1. Using the graphical view of the database, right-click the CUSTROOT segment. This brings up the options that are shown in Figure 11-22. Select **Import COBOL or PL/I Data Structures**.

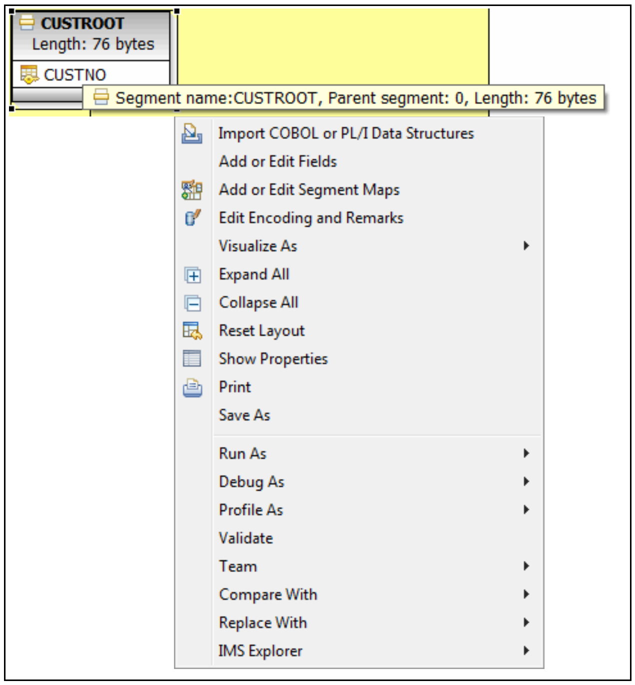


Figure 11-22 Import COBOL or PL/I Data Structures

2. The Import Data Structures window opens, as shown in Figure 11-23 on page 341. By using this window, you can import the COBOL copybook or PL/I Include that describes the database segment.

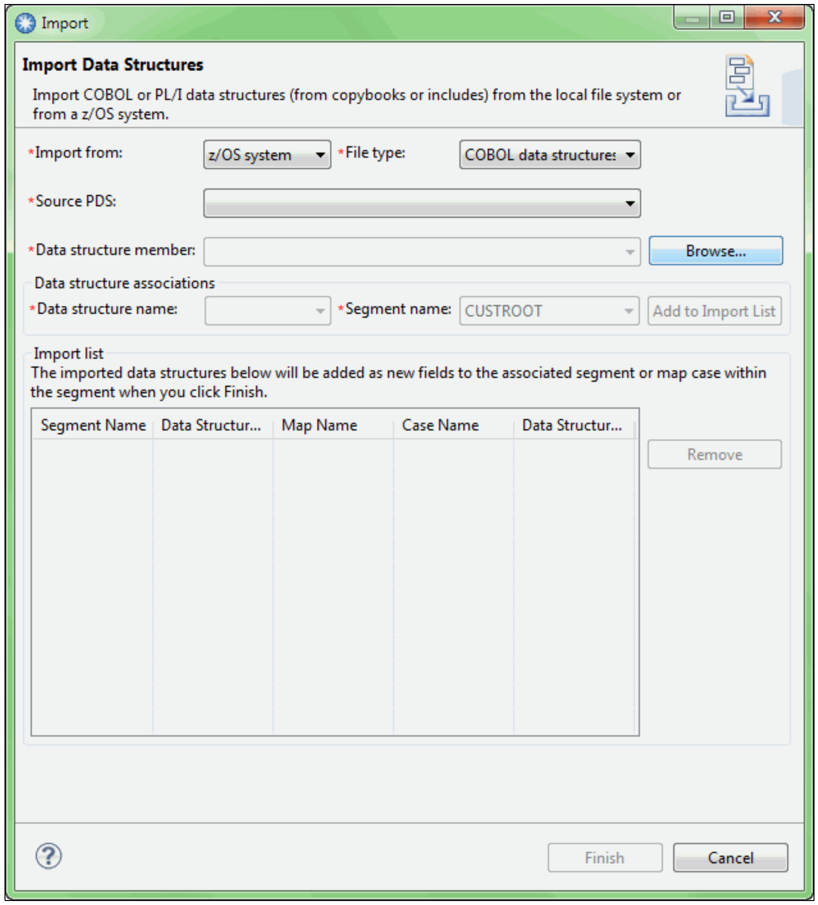


Figure 11-23 Import Data Structures window

3. We use the FTP connection that we set up earlier to access the source for the DBD. In Import From**,** select **z/OS System**, and then click **Browse** to select the data set that contains the data structure you want to import from the list that is shown in Figure 11-24.

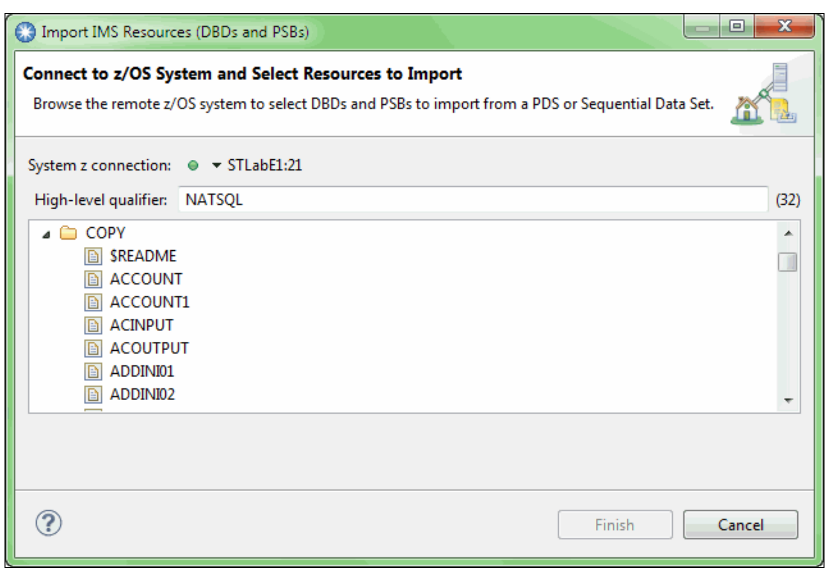


Figure 11-24 Select data set member that contains the data structure to be imported

4. Scroll down to the member you want to import (in this case, CUSTOMER, as shown in Figure 11-25). Click **Finish**.

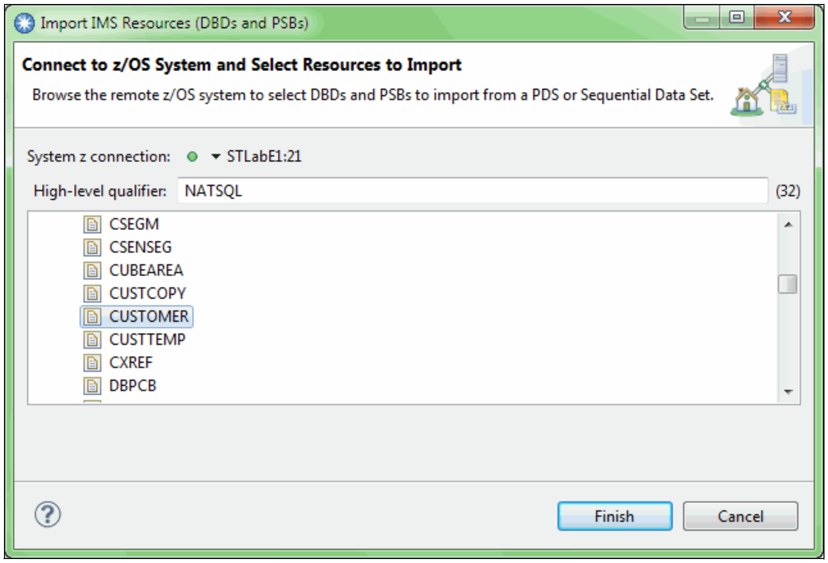


Figure 11-25 Select the library member to be imported

5. The wizard then looks through this member, identifying potential data structure names to be associated with the segment name. If there are several potential data structures in the member that is selected, choose the one you one that you want to use in the Data Structure Name list, as shown in Figure 11-26 on page 343. When the association is correctly matched, click **Add to Import List**.

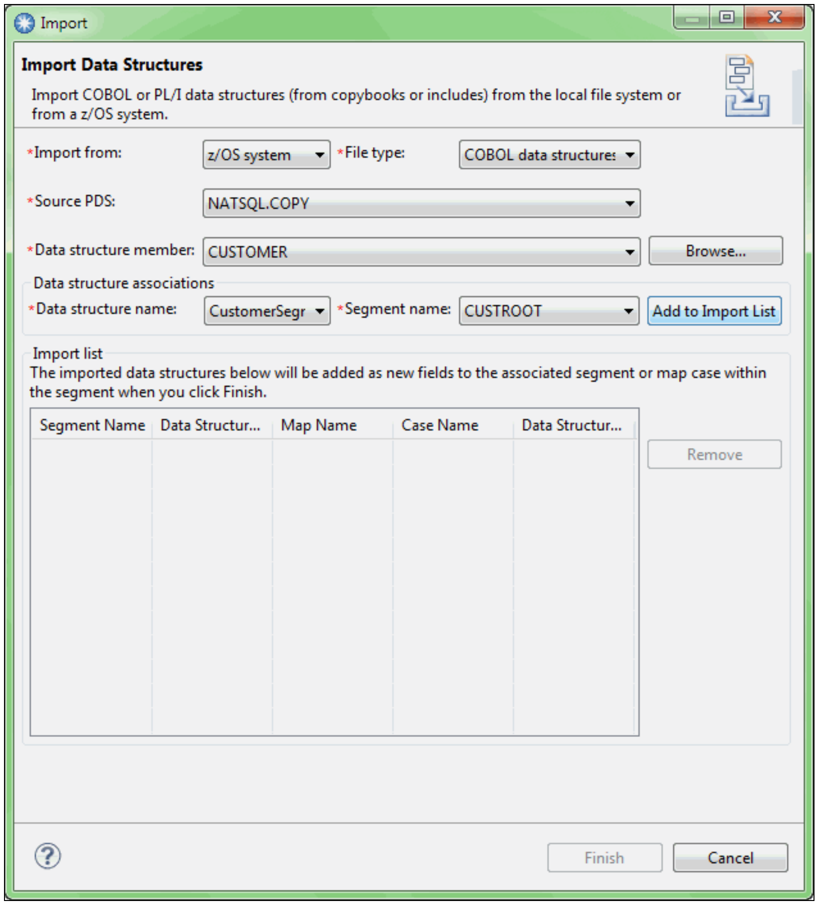


Figure 11-26 Associate the Data Structure Name with the Segment Name

6. The Import Data Structure window now shows the associated Segment Name and Data Structure Name in the Import List. Click **Finish** to get the wizard to import the fields in the Data Structure into the definition of the Segment in the DBD, as shown in Figure 11-27 on page 344.

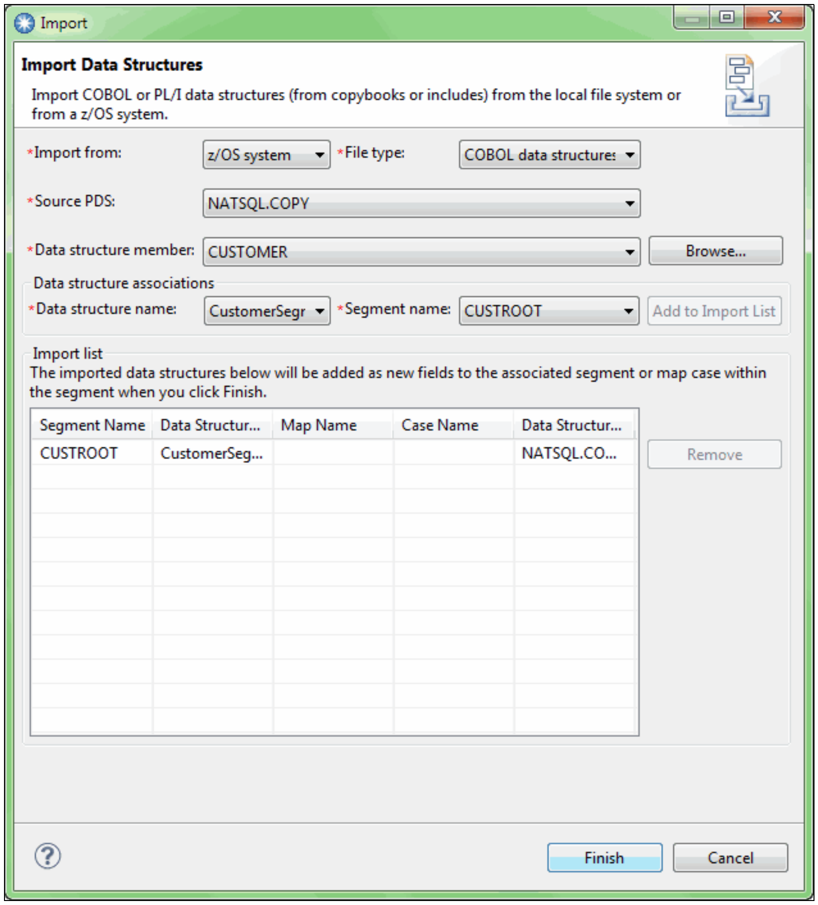


Figure 11-27 Import the fields in the Data Structure into the definition of the Segment in the DBD

The wizard shows the graphical display of the DBD, including the fields in the data structure that were imported, as shown in Figure 11-28.

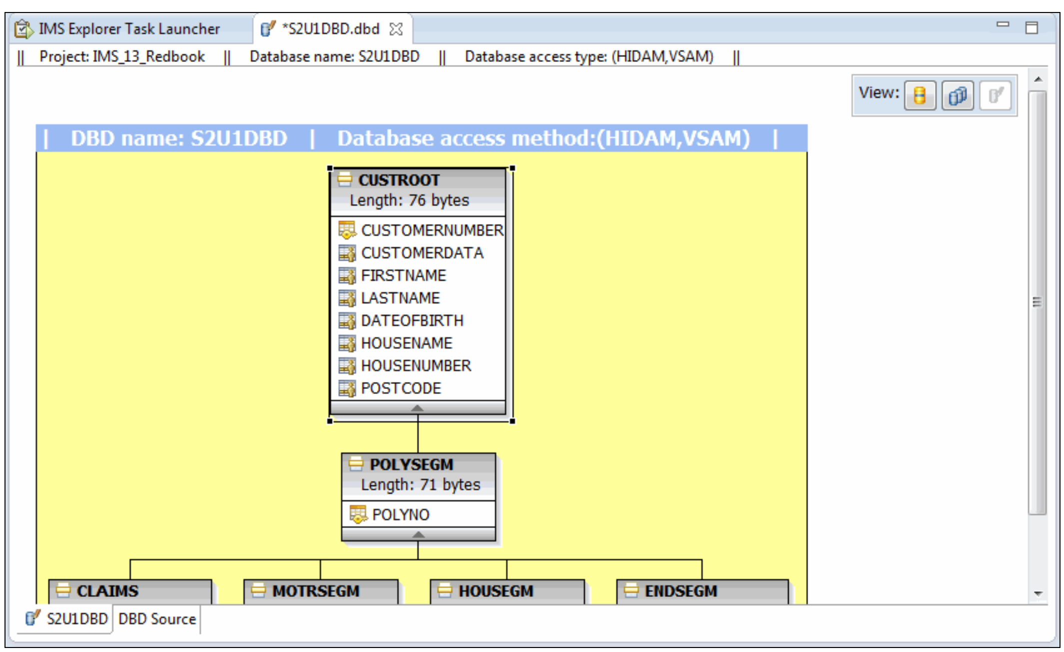


Figure 11-28 Customer segment, including the fields that are imported by the wizard

7. The expanded DBD source can be viewed by clicking the **DBD Source** tab at the bottom of this panel, as shown in Figure 11-29.

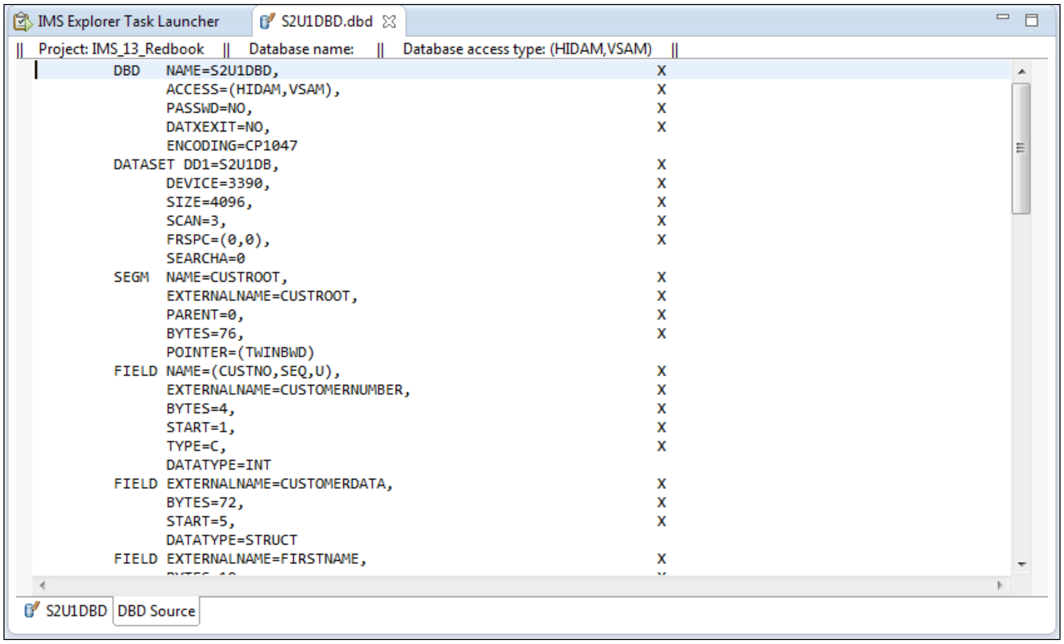


Figure 11-29 Expanded DBD source after the data structure is imported

8. Repeat this import process for each of the segment types in the database. Eventually, you have all segments in the database populated.

   Figure 11-30 on page 346 shows the database layout with definitions for each of the segments in the database. The asterisk at the left of the database name (*S2U1DBD.dbd) indicates that the database definition is not yet saved. Click **File** → **Save** to save the enhanced database definition to the project.

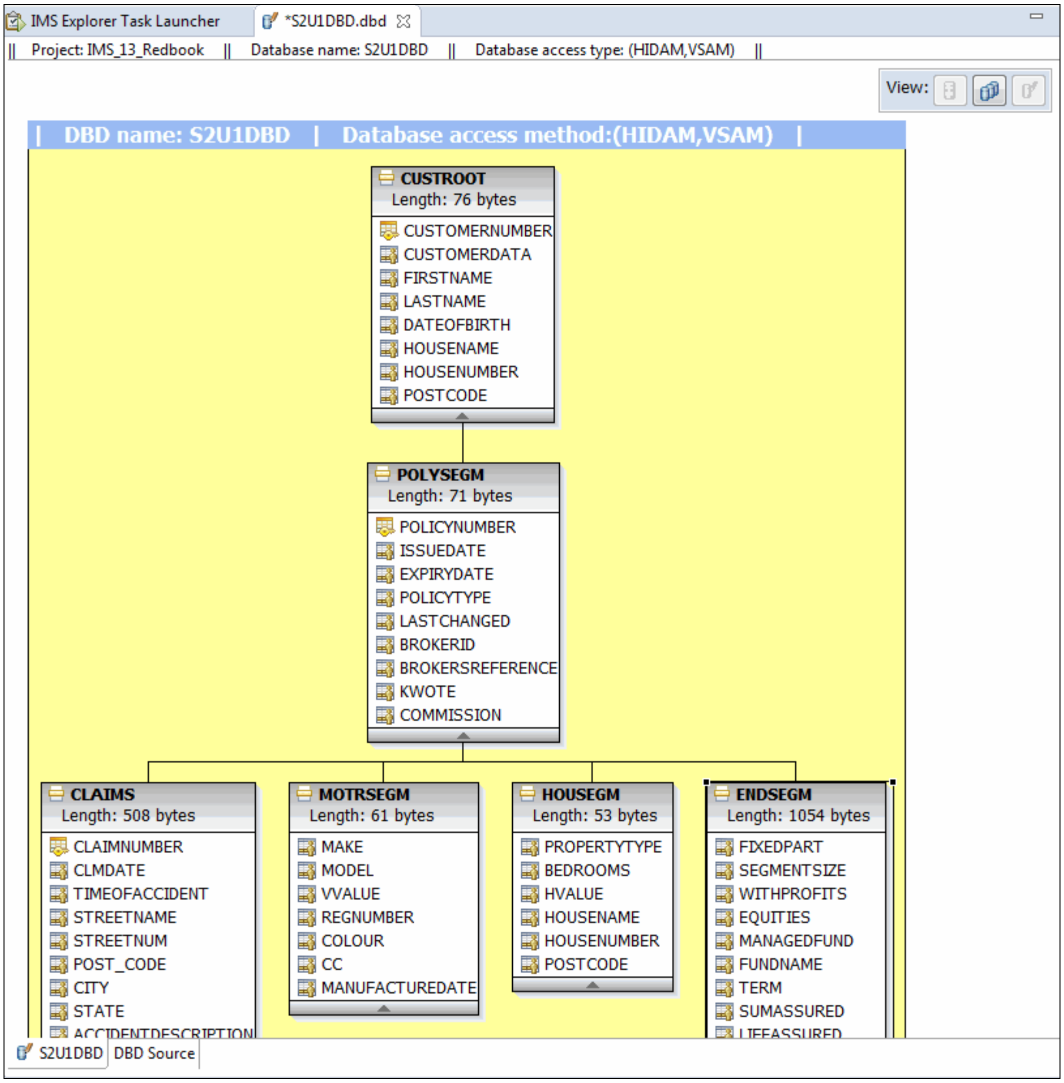


Figure 11-30 Database Layout, with all segment types populated

From here, you can use your existing IMS database definition processes to include the enhanced database definition into your IMS system. Complete the following steps to include the definition in your system:

1. Copy the expanded DBD source to the host.

   There are several ways to perform this copy. Recent versions of IBM Personal Communications include an FTP client with a graphical user interface (GUI). A simple file transfer is also available through your 3270 emulator.

2. Generate these DBDs with standard DBDGEN processes.
3. Perform an ACBGEN and Catalog Populate utility execution.

   The updated database description must be added into your ACB library and to the catalog. This process can be done in a single utility execution or separately.

The database metadata is now captured in the catalog. Existing applications continue to access their databases as they always did. New Java applications can use the catalog definitions to simplify the JDBC access to the same IMS databases.

11.4 Using IMS Explorer to Access IMS Data

The IMS Enterprise Suite can be used to access IMS databases through the SQL. This section describes how to create a database connection to IMS, create a Data Development Project for your SQL statements and scripts, and how to use the assistance that is built into IMS Explorer for Development to create SQL for both novice and experienced people.

11.4.1 Creating a database connection

Complete the following steps to create a database option:

1. Start z/OS Explorer, and open the IMS Explorer perspective, as shown in Figure 11-31.

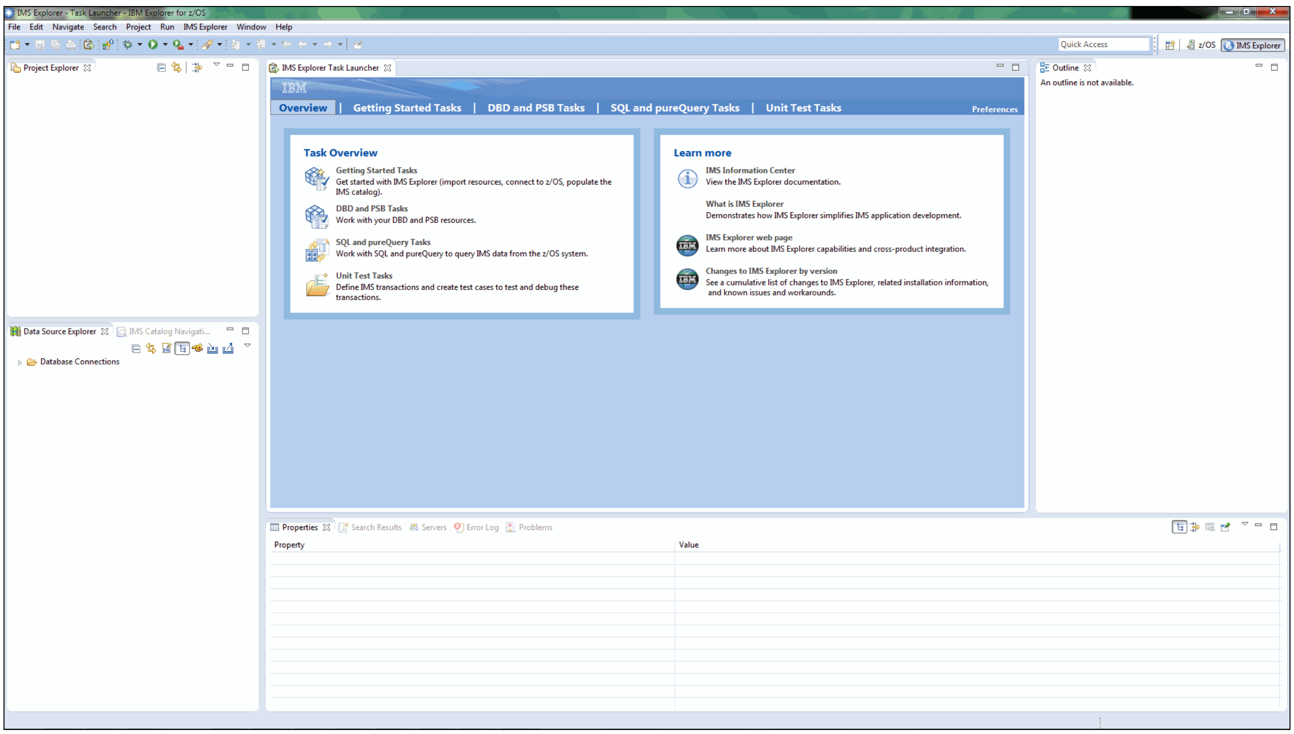


Figure 11-31 z/OS Explorer, showing the IMS Explorer perspective

2. In the Data Source Explorer pane (lower left in the default layout), right-click **Database Connections** and select **New...,** as shown in Figure 11-32.

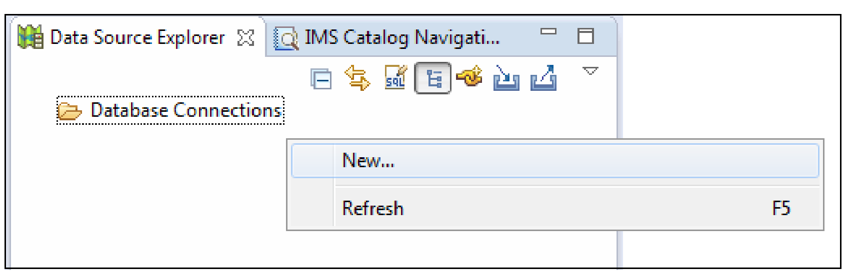


Figure 11-32 Setting up a new Database Connection

3. The New Connection window is shown in Figure 11-33. In Select a database manager, choose **IMS**. Ensure that the JDBC driver is IMS V13 Universal JDBC driver Default, and then enter the following parameters that are needed to connect to your IMS system:

   – Host name (or dotted address)
   – Port number for the ODBM connection
   – User name
   – Password
   – Source for the database metadata (usually the catalog)
   – PSB to be accessed

   Note: The host name and port number can be obtained from your IMS Connect address space by using the **VIEWHWS** command.

   When you entered all the parameters, the connection can be checked by clicking **Test Connection**. Before the connection is created, it might be useful to review the trace properties by selecting the **Tracing** tab in the properties panel.

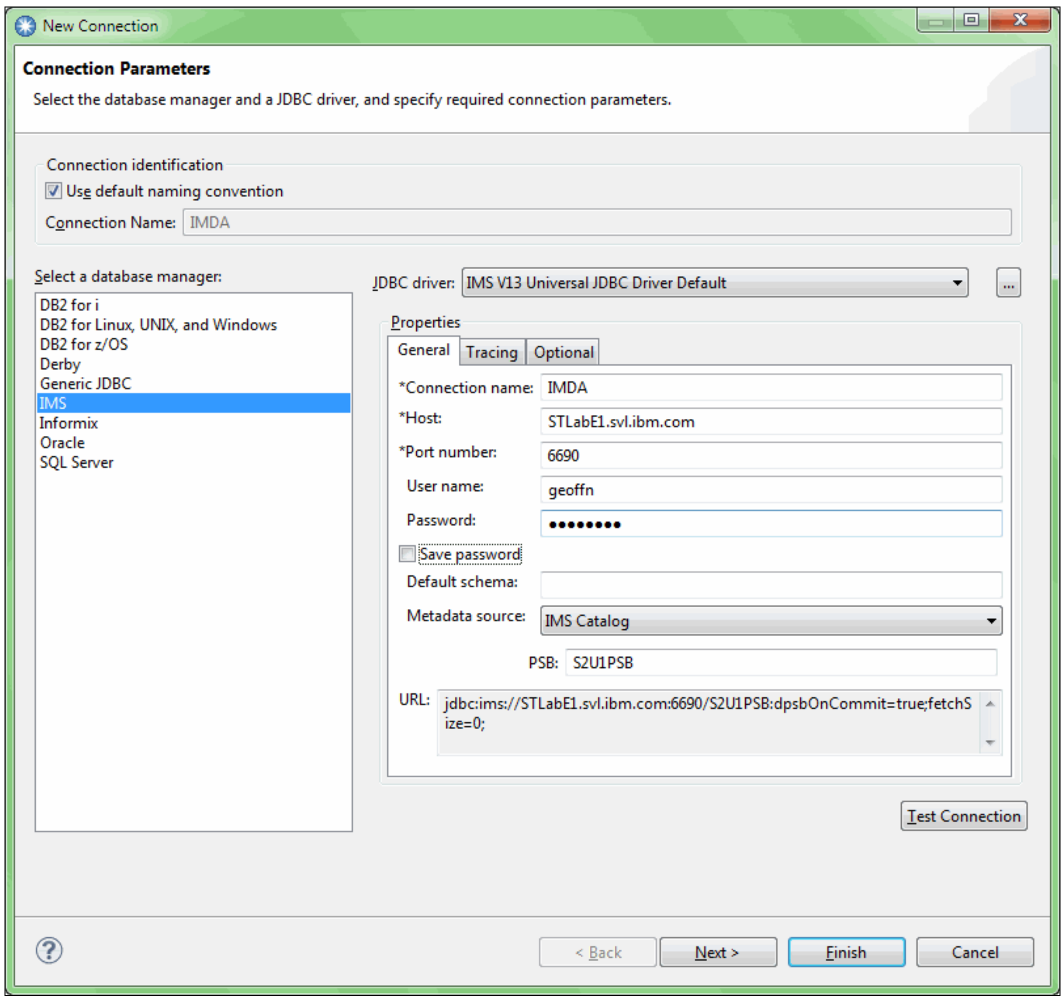


Figure 11-33 New Connection: Connection Parameters

4. By default, tracing on an IMS connection is disabled (as shown in Figure 11-34). To enable tracing, clear **Disable Tracing**, then enter the directory and file name on your workstation where the trace is to be saved. The wanted trace levels also can be selected. When complete, click **Finish**, as shown in Figure 11-33 on page 348.

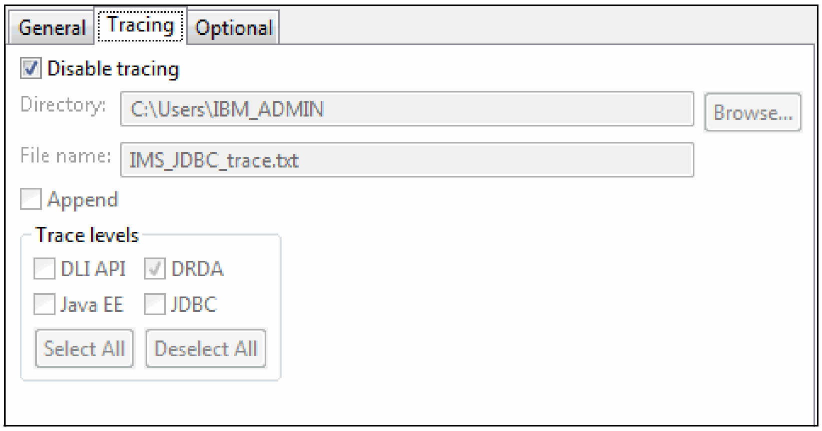


Figure 11-34 Tracing options for an IMS connection

5. The IMDA database connection is now populated by the wizard, as shown in Figure 11-35. The PSB that was specified in the connection is S2U1PSB, with the PCB (PCB01) expanded to show the tables (segments) that can be accessed. These tables can be further expanded to show the columns (fields) in each table.

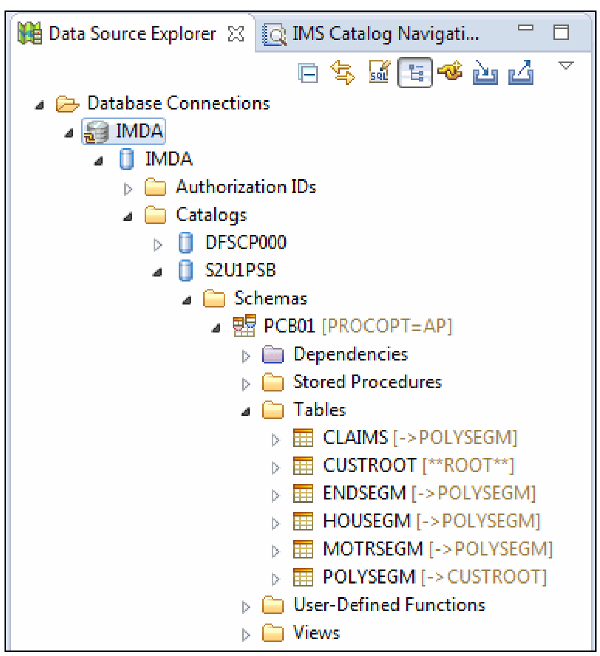


Figure 11-35 Expanded view of an IMS Database Connection

11.4.2 Sample Data by using a Database Connection

Complete the following steps to verify that everything is working as expected by sampling some data from the database:

1. Right-click the CUSTROOT table in the IMDA database connection in Data Source Explorer (as shown in Figure 11-36). Select **Data** → **Sample Contents**.

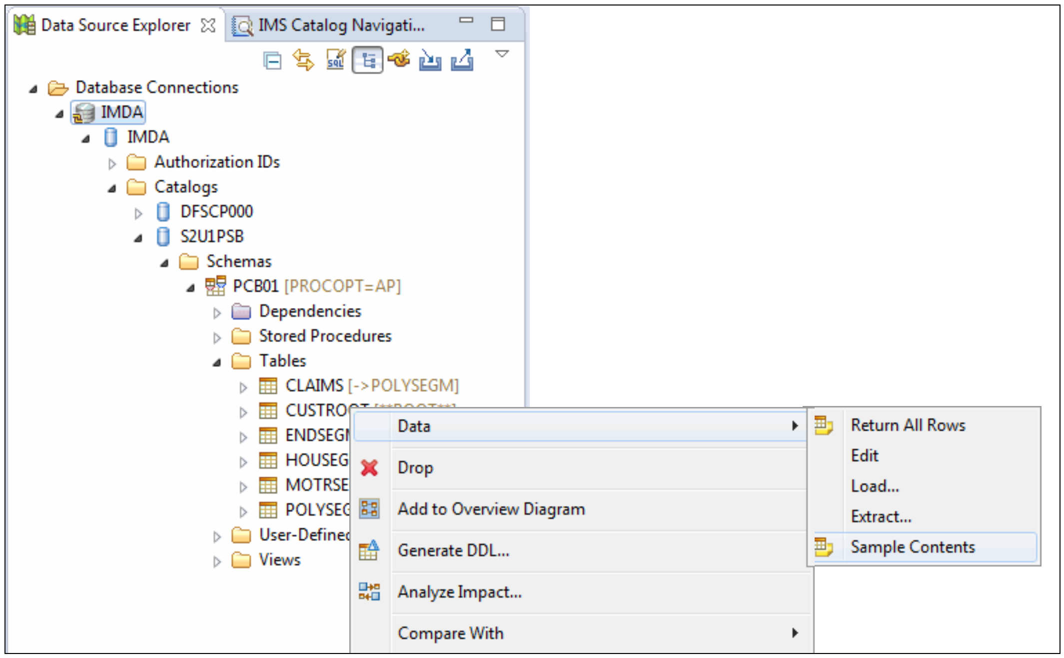


Figure 11-36 Sample the contents of an IMS database

2. IMS Explorer accesses the CUSTROOT table and returns the first 50 rows in the Result tab of the SQL Results panel, as shown in Figure 11-37.

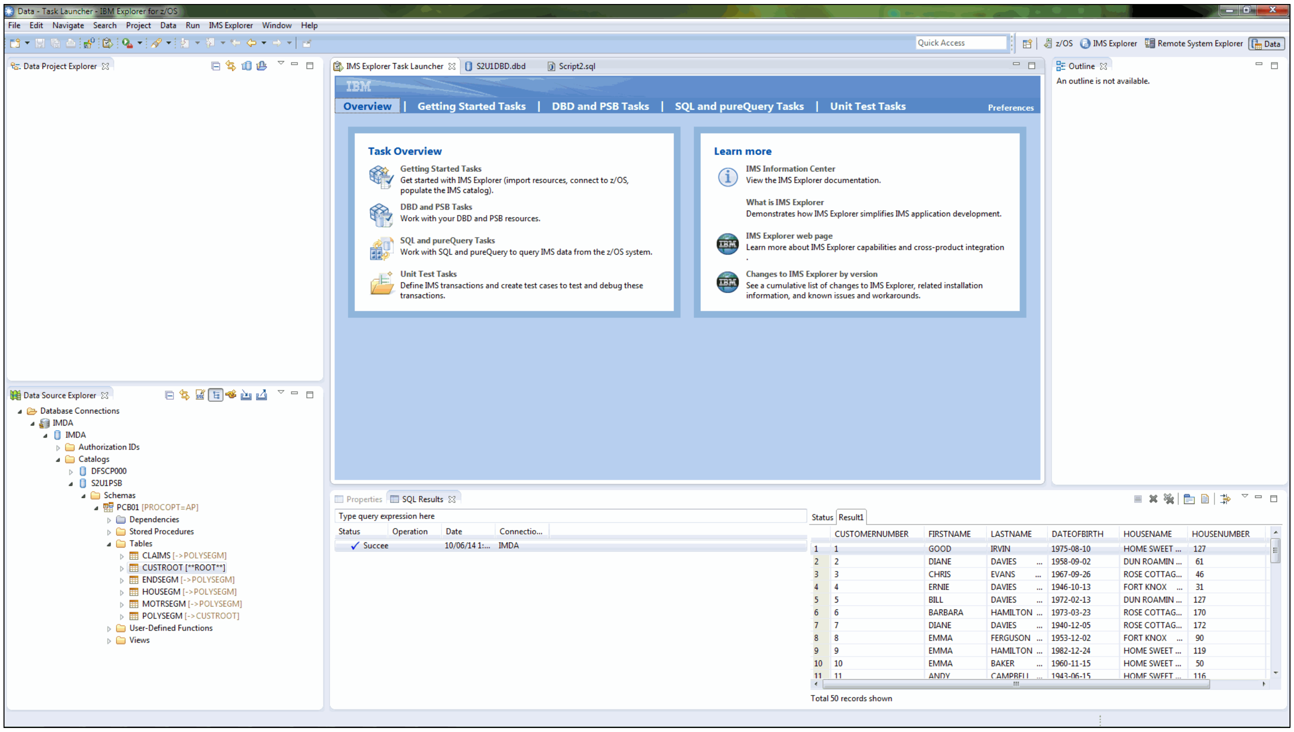


Figure 11-37 IMS Explorer perspective in z/OS Explorer

Details of the sample results are shown in Figure 11-38.

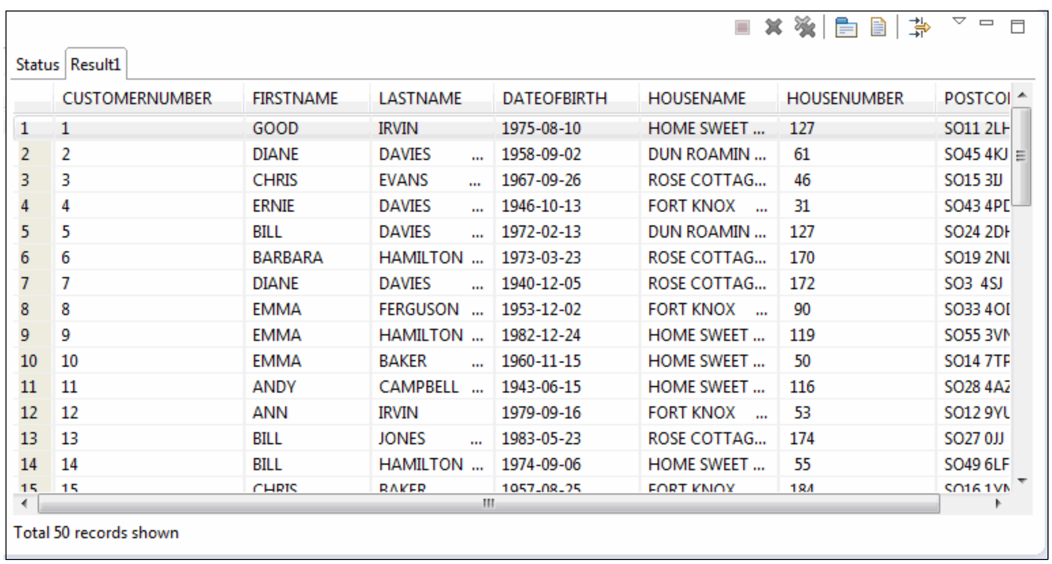


Figure 11-38 Results from sampling the IMS database

11.4.3 Creating a Data Development Project

To create your own SQL to access IMS data, a Data Development Project must be created. Complete the following steps:

1. Right-click in the Data Project Explorer pane, select **New** → **Data Development Project**, as shown in Figure 11-39.

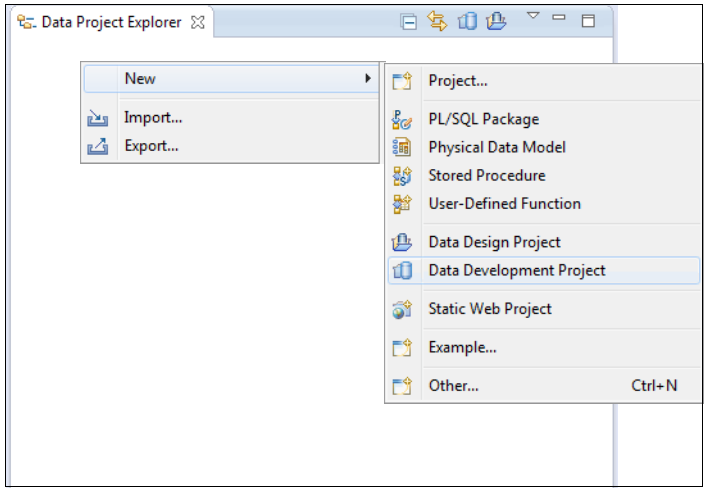


Figure 11-39 Creating a project in Data Project Explorer

2. In the New Data Development Project window (as shown in Figure 11-40), enter the name of the project, and then select **Next**.

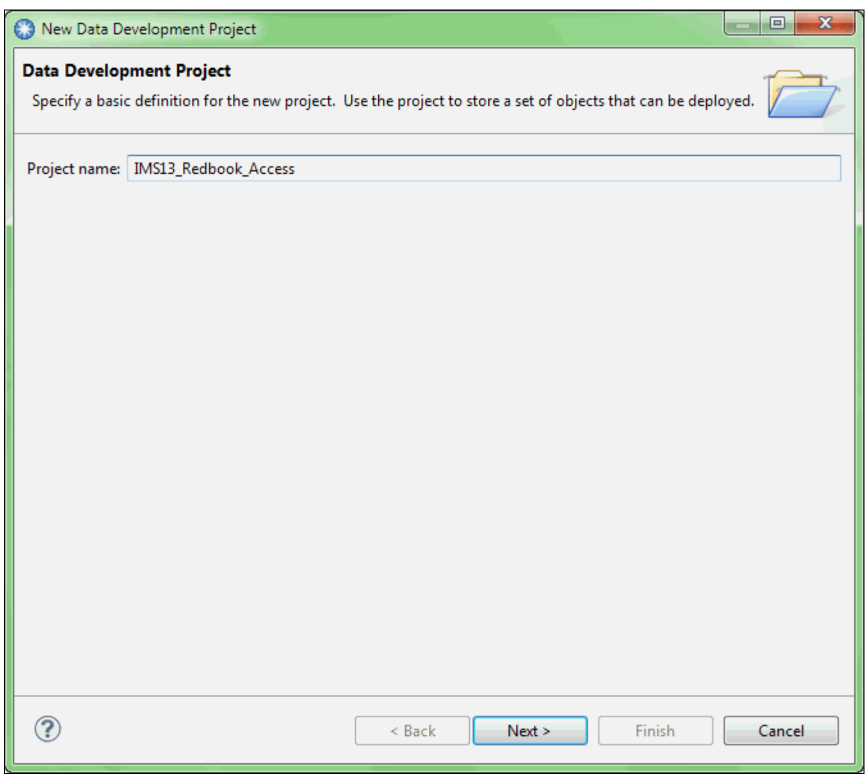


Figure 11-40 Data Development Project: Assign a project name

3. In the Select Connection pane (as shown in Figure 11-41), choose the connection to be used (in our example, IMDA). Click **Finish**.

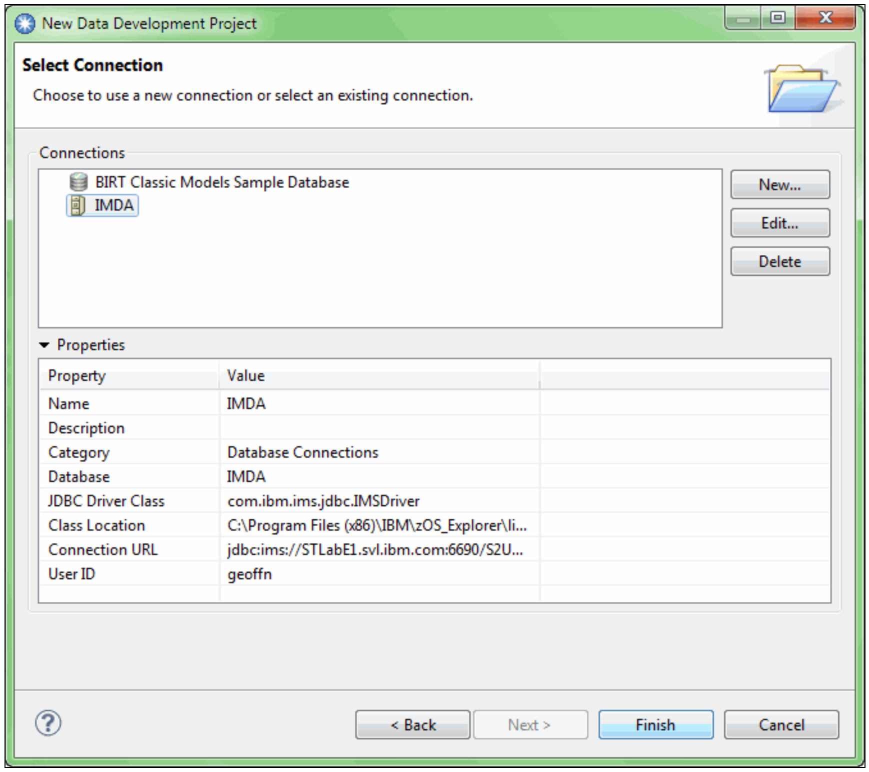


Figure 11-41 New Data Development Project: Select Connection

We now have a data access project (IMS13_Redbook_Access) in Data Project Explorer, as shown in Figure 11-42.

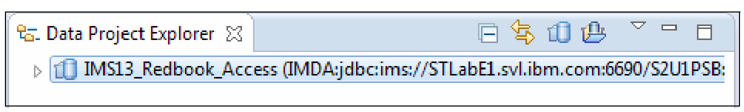


Figure 11-42 IMS13_ Redbook_ Access project in the Data Project Explorer pane

11.4.4 Creating an SQL or XQuery script

Complete the following steps to create an SQL or XQuery script:

1. Right-click the IMS13_Redbook_Access entry, then select **New** → **SQL or XQuery Script**, as shown in Figure 11-43 on page 354.

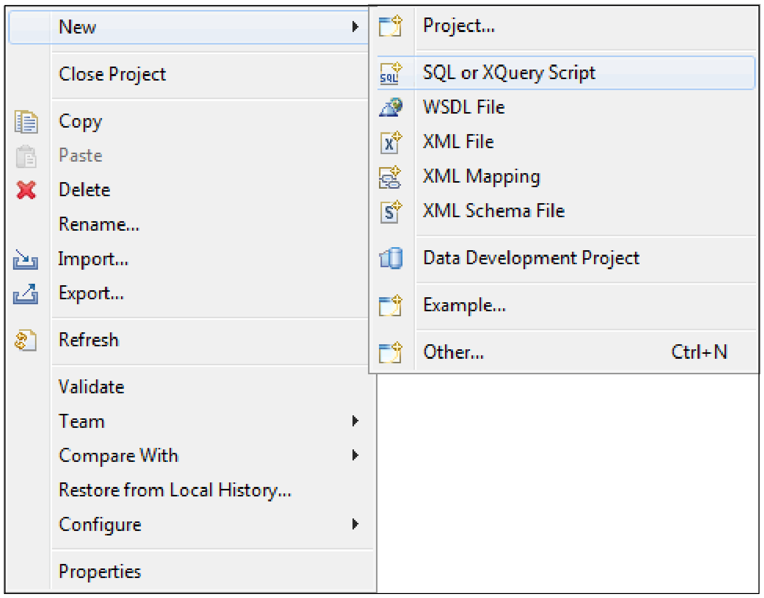


Figure 11-43 Create a New SQL or ZQuery Script

2. In the New SQL or XQuery Script window (as shown in Figure 11-44), enter the name of the script (in our example, Script1), select **SQL Query Builder**, and the statement type (in our example, SELECT). Click **Finish**.

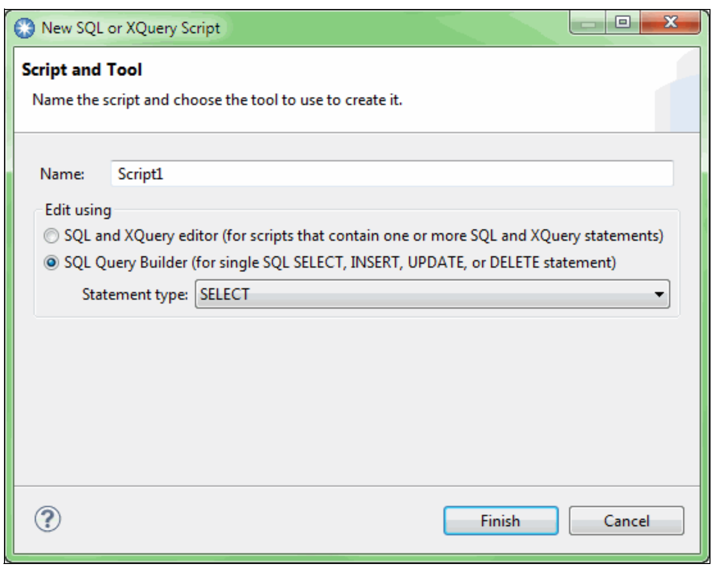


Figure 11-44 New SQL or ZQuery Script: Script and Tool

3. The IMS Explorer Task Launcher window opens, as shown in Figure 11-45 on page 355. From here, several options are available. If you are confident with your SQL skill, the SQL statement can be directly entered into the top pane.

   For users with less SQL experience, right-click in the middle pane and select **Add Table**.

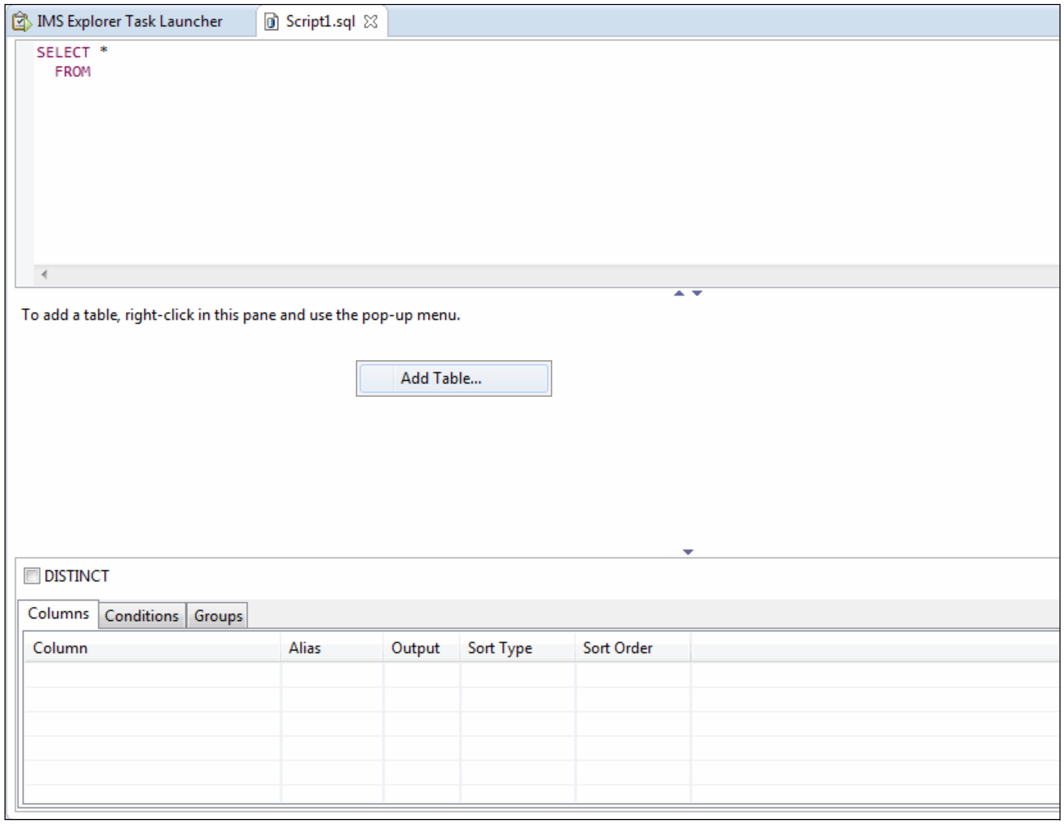


Figure 11-45 IMS Explorer Task Launcher

4. The Add Table window opens, as shown in Figure 11-46. The list of PCBs in the PSB that are being used for this data access project is shown. The first three PCBs are for the IMS catalog database and are implicitly part of every PSB. The PCB we use is PCB01; therefore, expand this PCB. The list of tables (segments) that are available with this PCB are shown. Select each of the tables to be accessed (use Ctrl+left mouse button to select multiple tables). When you select the tables to be used in the SQL statement, click **OK**.

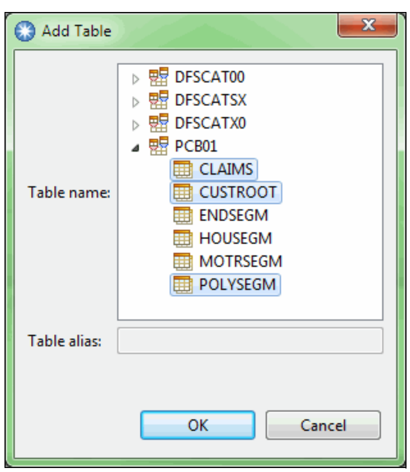


Figure 11-46 Add Tables to the SQL Query

5. The tables that you selected are shown in the middle section of the IMS Explorer Task Launcher, as shown in Figure 11-47. The SELECT statement in the upper pane now includes the qualified names of each of these tables as part of the SQL statement.

   The box that shows a table can be expanded by dragging the lower-right corner down or across the window.

   To include a column (field) into your SQL query, select the check box that is next to the field. The relationship between these tables is already known to IMS and does not need to be explicitly included in the SQL because of the hierarchic structure of IMS databases.

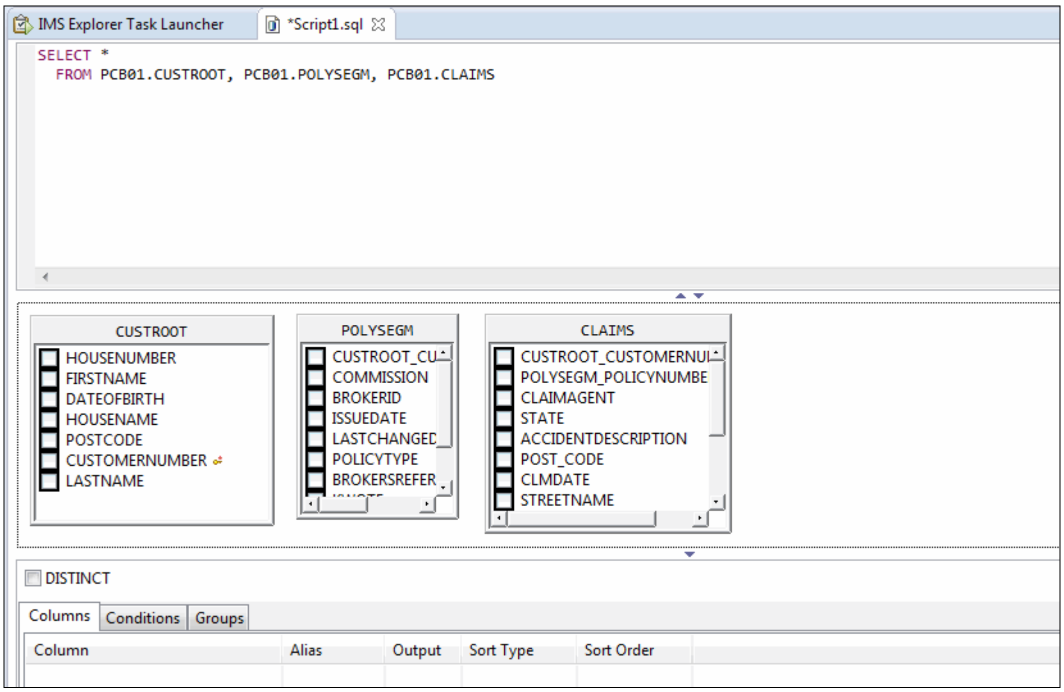


Figure 11-47 IMS Explorer Task Launcher: Tables

6. Figure 11-48 shows the IMS Explorer Task Launcher window when you have select the columns to be included into the SQL statement. The SQL statement is incrementally built as each column is selected.

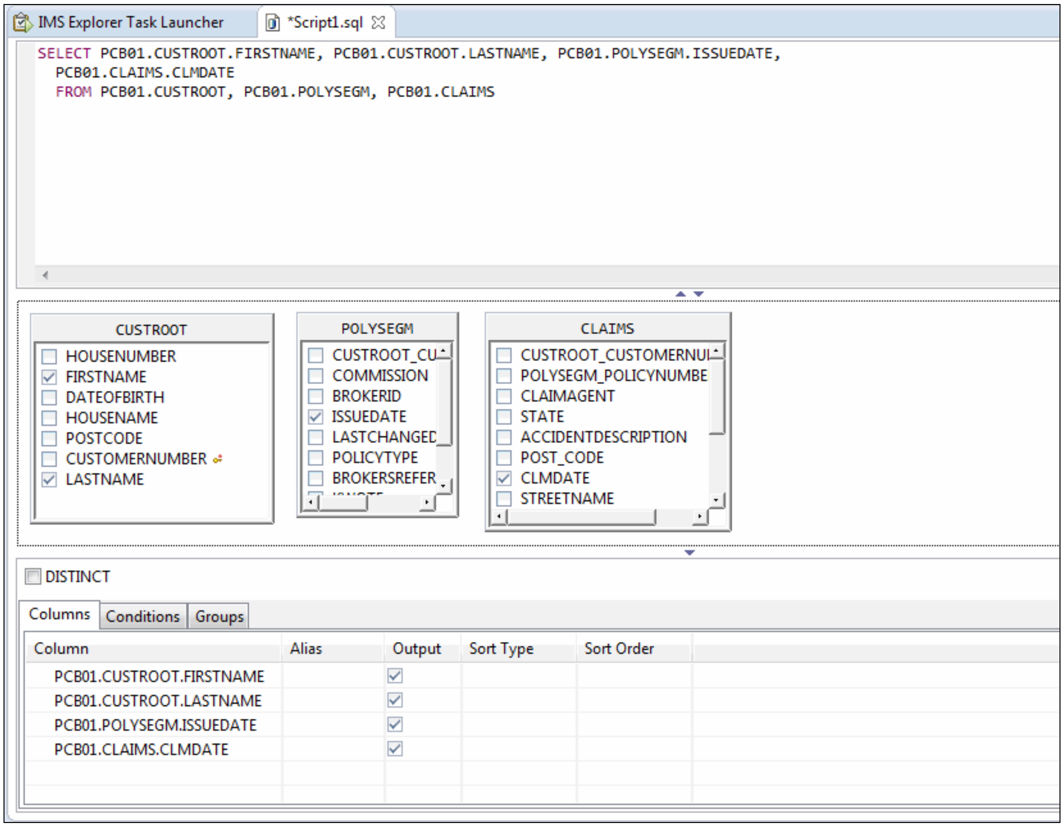


Figure 11-48 IMS Explorer Task Launcher: Columns Selected

7. To run the SQL, right-click anywhere in the top pane and select **Run SQL**, as shown in Figure 11-49.

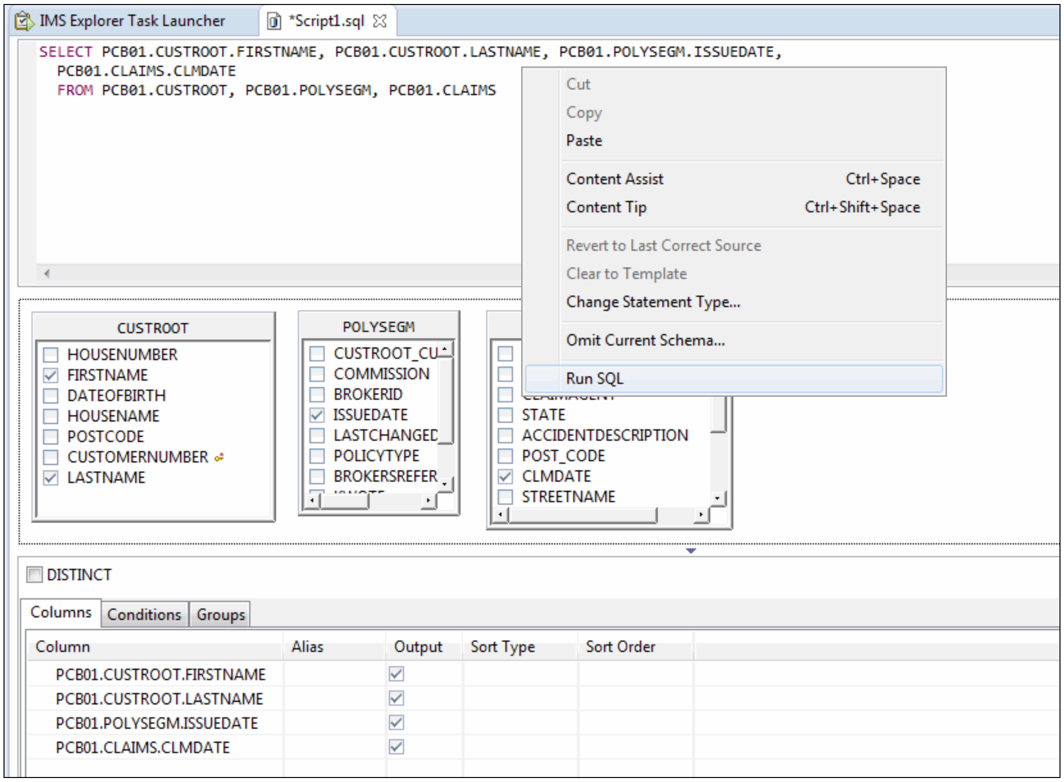


Figure 11-49 IMS Explorer Task Launcher: Run SQL

8. The results of the execution of the SQL are shown in the Result1 tab in the SQL Results pane in z/OS Explorer, as shown in Figure 11-50.

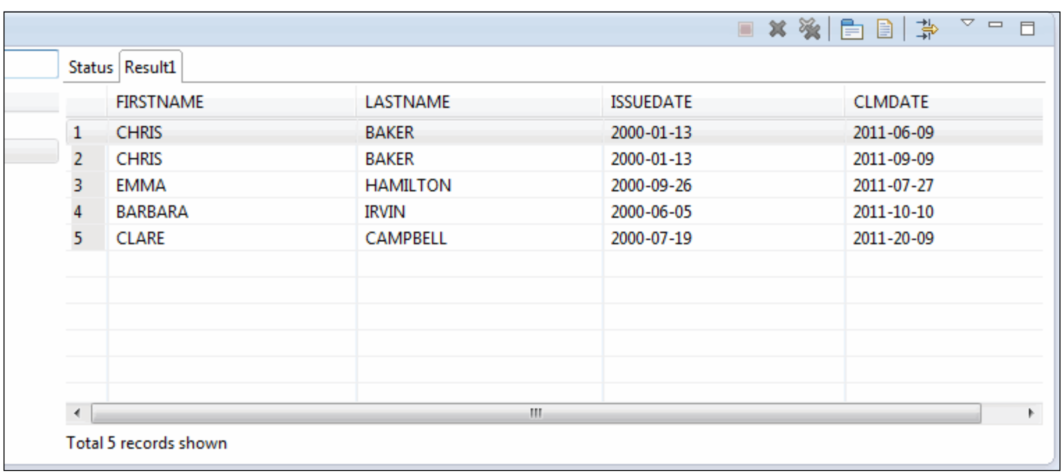


Figure 11-50 SQL Results

9. This SQL statement can be saved into your Data Development Project by selecting **File → Save** from the z/OS Explorer menu. The project can be used to save many SQL statements and scripts, which can be run again when needed, as shown in Figure 11-51 on page 359.

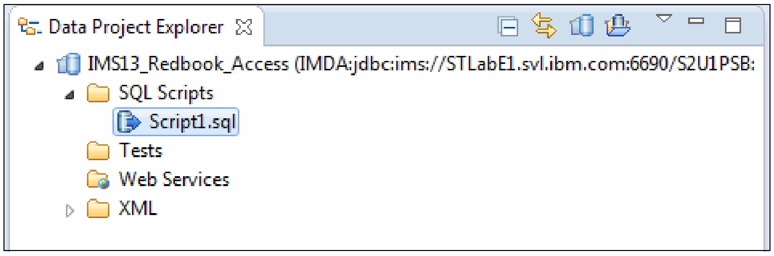


Figure 11-51 Data Project Explorer: SQL Scripts

11.5 IMS Enterprise Suite SOAP Gateway

MS Enterprise Suite Version 3.1 SOAP Gateway adds the support for 64-bit z/OS operating environments, send-only with acknowledge for synchronous callout, and SOAP Gateway management utility batch mode.

11.5.1 64-bit support for z/OS

SOAP Gateway now runs on the z/OS platform in 64-bit mode with which organizations can use their 64-bit operating environment for extended memory usage.

11.5.2 Send-only with ACK support for synchronous callout

Send-only with acknowledgment protocol support for synchronous callout is used by SOAP Gateway to receive a final confirmation that the response message was delivered to the original IMS application that issued the callout request. This confirmation provides SOAP Gateway users more information about whether a callout response message was sent to IMS and whether IMS received the message.

11.5.3 SOAP Gateway management utility batch mode support

Administrators can now use the batch mode of the management utility to facilitate web service deployment and server management for better performance and manageability. Instead of issuing one command at a time, each with its own JVM instance, you can pass a file with a list of commands to the SOAP Gateway management utility **`iogmgmt -batch`** command for execution as a batch in one JVM instance.

The batch command runs multiple SOAP Gateway management utility commands as a batch in one JVM instance.

11.5.4 Enhanced security cipher suite support

SOAP Gateway is enhanced to use the FIPS 140-2 approved cryptographic provider (or providers): IBMJCEFIPS (certificate 376) or IBMJSSEFIPS (certificate 409) for cryptography. The certificates are available at this website:

http://csrc.nist.gov/cryptval/140-1/1401val2004.htm

SOAP Gateway also adds the support for Transport Layer Security (TLS) V1.2 and for cipher suites with key length of 2048 and key strength of 112 bit, as required by NIST SP800-131a.

Federal Information Processing Standards (FIPS) are standards and guidelines that are issued by the United States National Institute of Standards and Technology (NIST) for federal government computer systems. FIPS can be enabled for SOAP Gateway.

11.6 IMS Data Provider for Microsoft .NET

With the IBM IMS Data Provider for Microsoft .NET, you can use standard SQL queries to access IMS data from .NET applications. For more information about this function, see *IMS Integration and Connectivity Across the Enterprise,* SG24-8174.

11.6.1 Overview of IMS Data Provider for Microsoft .NET

The IMS Data Provider for Microsoft .NET provides the solution for Microsoft .NET-based applications to access and manipulate IMS data. It is based on the ADO.NET specifications to handle the connections and communications with IMS through IMS Connect by following the Distributed Relational Database Architecture (DRDA) protocol.

.NET application developers can use their preferred development environment, such as Microsoft Visual Studio, to call the provided APIs to access IMS databases by adding the IMS Data Provider for Microsoft .NET DLL library file in the reference path of their client applications.

ADO.NET

ADO.NET is the Microsoft ActiveX Data Object. It is an architecture that provides access to data from distributed, data sharing applications. The approach that is used by ADO.NET to access the data is independent of the underlying data store, whether it is a DBMS, an XML document, or application data. Therefore, the same programming model can be used to access each of these different types of data.

Programming languages

The programming languages that are commonly used to write application programs to access data that uses .NET include C# (C sharp) and Visual Basic. Other ADO.NET compliant languages can be used, including F#, and C++. The IDE that was used to create these applications is Microsoft Visual Studio. We provide sample code in C# to connect to and access data from IMS databases in 11.6.3, “Making a connection to IMS” on page 361.

11.6.2 Setting up IMS to communicate with .NET

There are several things that must be in place for a .NET application to access IMS data. First, TCP/IP connectivity to your IMS must be available through IMS Connect. Access to the data through IMS Connect is provided by the Open Database Manager (ODBM) address space; therefore, ODBM and the Common Service Layer are also needed. Finally, the metadata for the IMS databases that are being accessed must be available in the IMS Catalog, as shown in Figure 11-52 on page 361.

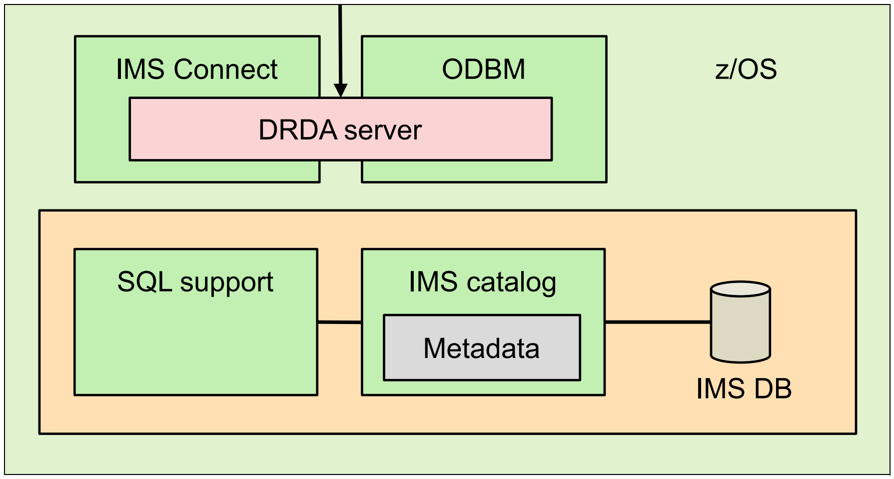


Figure 11-52 IMS components that are needed for .NET

For more information about importing metadata into the IMS catalog, see “Using IMS Explorer to capture IMS metadata” on page 328.

What is delivered by the IMS Data Provider for .NET

Every Data Provider for .NET delivers a common set of facilities and functions. These are implementations of the following basic classes:

- Connection

  Establishes and manages a database connection.

- Command

  Runs an SQL statement against a database.

- DataReader

  Reads and returns the result set from a database.

- DataAdapter

  Links a DataSet Instance to a database. Through a DataAdapter instance, the .NET application can read and write database table data.

Structure of a .NET application

The typical structure of a simple .NET application includes two parts. The first part is the `App.config` file that contains static information that is related to the application. This static, or parameter, information includes the connection string parameters that are needed to make your connection to IMS, switches to enable diagnostics and tracing, and any other parameters that you need in your application.

The second part is the executable code, which is written in a .NET language, such as C# or Visual Basic. The examples that are used in this book are written in C#.

11.6.3 Making a connection to IMS

In a .NET application, the connection to your IMS is achieved in two steps. This first step is to create a `connectionString`. This connection string, which is often stored in the `App.config` (as shown in Example 11-1 on page 362) contains the parameters that are needed to identify the IMS system and database (PSB) you are to access.

Example 11-1 Sample .NET Application: App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
    </configSections>
  <system.diagnostics>
    <switches>  // see "System Diagnostics" on page 365
      <add name="IMSDataProviderTrace" value="4"/>
    </switches>
  </system.diagnostics>
    <connectionStrings>
      <add name="IMSA"  // see "Connection Strings" on page 365
            connectionString="Server=zos.server,6690; Datastore=IMSA;
                              UID=myuserid; PWD=mypwd;"
            providerName="IBM.Data.IMS" />
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
  <appSettings>  // see "appSettings" on page 365
    <add key="SQLA11-PSB"  value ="S2U1PSB"></add>
    <add key="SQLA11-SQL"  value ="SELECT customernumber FROM
                                   PCB01.custroot"></add>

  </appSettings>
</configuration>
```

In the executable code, `Program.cs` (the statement that is shown in Example 11-2) makes the connection by using this connection string. The example includes links to sections in this chapter that provide more information about the code in the example.

Example 11-2 Sample .NET Application: Program.cs

```
 // see "using <namespaces>" on page 365
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Diagnostics;
using IBM.Data.IMS;
using System.Linq;
using System.IO;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
    // see "MyConnection" on page 365
        private static String MyConnection = "IMSA";
        private static Int16  DisplayCount = 10;
        private static String[] TestNameList = new string[1] { "SQLA11", };
        private static String TestName;
        private static Int16 TestNumber = 0;
```

```
        static void Main()
        {

            TestName = TestNameList[TestNumber];
            Console.WriteLine(TestName);
            Console.WriteLine("");
        // see "Set PCBName and MySQLStatement" on page 365
            string PCBName =
                   System.Configuration.ConfigurationManager.AppSettings
                    [TestName + "-PSB"];
            Console.WriteLine("PCBName: " + PCBName);
        string MySQLStatement =
                    System.Configuration.ConfigurationManager.AppSettings
                     [TestName + "-SQL"];
            Console.WriteLine("SQLStatement: " + MySQLStatement);

            // see "Set up tracing for the application" on page 365
            // Set tracing on, to file TestName_Date_Time_ClientLog.txt
            String MMMdd = DateTime.Now.ToString("MMMdd_HHmm");
            FileStream objStream = new FileStream("c:\\Log\\" + TestName + "_"
             + MMMdd + "_ClientLog.txt", FileMode.Create);
            TextWriterTraceListener objTraceListener = new
                      TextWriterTraceListener(objStream);
            Trace.Listeners.Add(objTraceListener);
            // forces trace to be flushed after every write
            Trace.AutoFlush = true;

            // Get the connection string from the app.config file
        // see "Define your Connection String" on page 365
            String connectionString =
                    ConfigurationManager.ConnectionStrings
                      [MyConnection].ConnectionString
                    + "Database=" + PCBName + ";";
            Console.WriteLine("after connectionstring:" + connectionString);
            Console.WriteLine("about to define a connection");

            // Create and open the connection in a using block. This ensures that
      // all resources will be closed and disposed when the code exits.

            using (
        // see "Define the Connection to IMS" on page 365
                IMSConnection connection = new IMSConnection(connectionString))
{

                Console.WriteLine("inside the using for the IMS Connection");
                Console.WriteLine("about to CreateCommand");
        // see "Create a Command" on page 366
        // Create the Command object.
                IMSCommand cmd = connection.CreateCommand();
                cmd.CommandText = MySQLStatement;

                // Open the connection in a try/catch block.
                // Create and execute the DataReader, writing the result set to
               // the console window.
                try
```

```
                {
                    Console.WriteLine("about to open connection");
                // see "Open the connection to IMS" on page 366
                    connection.Open();
                    Console.WriteLine("about to ExecuteReader cmd");
                // see "Execute the SQL statement" on page 366
                    IMSDataReader reader = cmd.ExecuteReader();

                    short Counter = 0;
                // see "Retrieve the data" on page 366
                    while (reader.Read() && Counter < DisplayCount)
                    {
                        for (int i = 0; i < reader.FieldCount; i++)
                        {
                            Console.Write(reader.GetName(i) + ": " +
                                          reader.GetString(i) + " ");

                            if (i == reader.FieldCount - 1)
                            {
                                Console.WriteLine("");
                                Console.WriteLine("");
                                Counter++;
                            }
                        }
                    }

                    Console.WriteLine("about to close reader");
              // see "Close the reader and connection" on page 366
                    reader.Close();
                Console.WriteLine("about to close connection");
                    connection.Close();
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Data);
                    Console.WriteLine(ex.Message);
                    Console.WriteLine(ex.StackTrace);
                    Console.WriteLine(ex.TargetSite);
                }

                Console.ReadLine();

                // flush any remaining traces, remove listener and close stream
            // see "Close the program trace" on page 366
                Trace.Flush();
                Trace.Listeners.Remove(objTraceListener);
                objStream.Close();
            }
        }
    }
}
```

Annotation of the sample .NET application

The following section describes the parts of the sample .NET application that is shown in Example 11-2 on page 362.

System Diagnostics

The `system.diagnostics` section enables the capture of trace information. The IMS Data Provider provides five levels of debug tracing. In this program, the tracing is set to verbose (value = 4), to learn as much as we can about the interactions between our application and IMS.

Connection Strings

The `connectionStrings` section provides a place for you to store the parameters that are used to connect to your databases. In this example, we have a single connection string (IMSA), with the many of the parameters that are required to access this IMS. There are several other parameters included in the connection string as the program is run.

appSettings

The `appSettings` section provides an area of static parameters to be stored for your application. There are two settings in this section: the name of the PSB and the SQL statement that is to be run.

using <namespaces>

The `using <namespaces>` statements allow your program to use classes that are provided by .NET and other providers. The `using IBM.Data.IMS` statement includes the classes that are provided by IBM to connect to and access IMS databases.

MyConnection

This statement defines a String variable and assigns it a value of "IMSA". It is used to identify the connection string that we defined in the `connectionStrings` section of `App.config`.

Set PCBName and MySQLStatement

These two statements extract the PCB name and SQL statement from the `App.config` file and store them in the string variables `PCBName` and `MySQLStatement` variables.

Set up tracing for the application

These statements create a file and requests that this file is used for tracing the diagnostic information for our program. These statements are not necessary, but the trace information can be valuable in the early stages of writing your first .NET application.

Define your Connection String

The `String connectionString` statement performs the following tasks:

- Defines the string variable that is about to be used to connect to IMS.
- Extracts the connection string that was set up in `App.config` and puts it into this variable.
- Appends the name of the database (PCB) to be accessed by this program.

With this statement, we defined all the parameters that are needed in the connection string when we connect to IMS.

Define the Connection to IMS

As with many statements in this program, various tasks are accomplished here. A new variable `IMSConnection` (of type connection) is defined. A new instance of a `connection` is created by using the values that are stored in variable `connectionString`.

In the variable connectionString, we defined the parameters to be used when we open the connection from our application to our target IMS.

Create a Command

The first of these two statements creates the IMS command object **`cmd`**. The SQL statement to be executed, which was stored in MySQLStatement, is assigned to the **`CommandText`** attribute of this **`cmd`** variable.

Open the connection to IMS

The **`Open`** method of the **`connection`** class is executed, which connects the application program to the IMS identified with the connection string.

Execute the SQL statement

The **`ExecuteReader`** method of the **`cmd`** class is run, which executes the SQL we set up. The database is read through the connection and IMS creates a result set that consists of the data in the database to be returned to the program. This result set is accessed through the **`Reader`** variable of type **`IMSDataReader`**.

Retrieve the data

The **`Read`** method of the **`reader`** variable performs several tasks. When it is started, it returns **`true`** for the next row in the result set. When a **`false`** value is returned, there are no more rows in the result set. The **`while`** statement is run for each row in the result set (up to a maximum of DisplayCount).

Each invocation of the **`reader.Read()`** method also returns the next row of data from the result set. The column name (**`reader.GetName(i)`**) and its value (**`reader.GetString(i)`**) are written to the console in each iteration of the **`for`** loop. The number of fields in each row is the attribute **`Reader.FieldCount`**.

Close the reader and connection

The **`reader.Close`** and **`connection.Close`** methods are started to close the command reader and the connection to IMS.

Close the program trace

The last task to perform is to close off the program trace.

12

Tools for IMS 13

IBM continues to increase the investment in tools to deliver comprehensive solutions that respond to customer requirements faster than ever before.

The focus of the investment is on common interfaces, cross-tool integration, and usage testing to enhance the performance of your IMS.

This chapter provides an overview of the IMS Tools portfolio and of the major updates to the tools to support IMS 13. We highlight the IBM IMS Tools strategy through solution packs, including descriptions, documentation, maintenance, and compatibility with various IMS versions.

This chapter includes the following topics:

- IMS Tools general information
- IMS Tools useful links
- IBM Tools Base for z/OS V1.4
- IMS Fast Path Solution Pack for z/OS V1.3
- IMS Database Solution Pack for z/OS V2.1
- IMS Recovery Solution Pack for z/OS V1.1
- IMS Performance Solution Pack for z/OS V1.2

12.1 IMS Tools general information

Designed to optimize data across your enterprise, IMS Tools deliver the reliability and affordability you need to help increase the value of your IMS environment.

The tools support each new version of IMS as it becomes available, making it easy for you to migrate to the latest version while continuing to benefit from tools support.

12.1.1 Using IMS upgrades

Upgrading to IMS 13 features the following benefits:

- Highest efficiency, lowest total cost of ownership
- Openness with industry standard interfaces
- Faster time to market with Database Versioning
- Increased availability, autonomics, and dynamic capabilities
- Integration
- Usability and simplification
- IMS Enterprise Suite
- IMS Explorer for Administration
- IMS and Big Data

12.1.2 Why IBM IMS Tools

IBM is current in its support of new features in IMS with usage and toleration. With IBM tools, you can migrate to new versions of IMS that are based on your business needs. There is an ongoing IBM investment beyond IMS support. New releases add features and functions to solve business challenges. IBM IMS, Utilities, and tools increase stability and reduce technical support interactions.

12.1.3 IBM IMS Tools support areas

IBM IMS Tools primarily deliver support and assistance in the following areas:

- Database administration and change management
- Utilities management
- Backup and recovery
- Performance management
- System and transaction management
- Data replication
- Testing and migration management
- Information governance

12.1.4 IBM IMS Tools Solution Packs

To meet the objectives about common interface and cross-tool integration, IBM announced the following Solution Packs (product numbers) in 2010:

- IBM Tools Base for z/OS (5655-V93)
- IMS Fast Path Solution Pack for z/OS (5655-W14)
- IMS Database Solution Pack for z/OS (5655-DSP)
- IMS Recovery Solution Pack for z/OS (5655-V86)
- IMS Performance Solution Pack for z/OS (5655-S42)

With the IMS Tools Solution Packs, the solutions you need are consolidated and integrated into a single, lower-cost offering. These offerings provide outstanding return-on-investment benefits and help reduce total cost of ownership while facilitating preferred practices and enabling growth. The following benefits are realized:

- Uses IMS 13 performance savings immediately
- Automates ongoing IMS database monitoring and maintenance
- Provides insight into the health and availability of IMS databases
- Enhances application programmer productivity

You can still order some of the products that are included in these packs individually, but most of them are now available only through those packs.

Other products, which are not included in packs, can be ordered as before.

This section provides an overview of those tools and the major updates that were made to some of them to support IMS 13.

12.1.5 Focusing on performance tooling

The major focus of IMS 13 is performance. Although IMS made extraordinary strides, your results are specific to your environment.

By using the performance tools, you can evaluate performance trends by comparing the following values:

- Last week or last month
- Previous version of IMS
- Previous version of DB2
- Previous hardware installation

12.2 IMS Tools useful links

The information that is available on the Internet is constantly updated. Check the referenced websites in the following sections periodically to ensure that you have the latest version of the information.

12.2.1 IBM DB2 and IMS Tools

The IBM DB2 and IMS Tools enhance the performance of IMS and DB2. The tools were upgraded and enhanced to work with IMS 13.

For more information about the tools, including the IMS versions that they support, see the DB2 and IMS Tools for System z page at this website:

http://www.ibm.com/software/data/db2imstools

12.2.2 IBM IMS Tools

IBM IMS Tools provide on-demand, application-specific database management solutions to enhance the performance of your Information Management system.

For more information about all IMS Tools, see the IBM IMS Tools page at this website:

http://www.ibm.com/software/data/db2imstools/products/ims-tools.html

12.2.3 IMS Tools product documentation

This library provides links to IMS Tools publications, program directories, and other related technical content. These documents are available in PDF, HTML, and BookManager® formats.

For more information about IMS Tools documentation, see the IMS Tools Product page, which is available for DB2 and IMS Tools at this website:

http://www.ibm.com/support/docview.wss?uid=swg27020942

12.2.4 IMS Tools and IMS 13 compatibility

For more information about the minimum maintenance level that is required for IMS Tools to support all the current IMS versions (10, 11, 12, and 13), see the IMS Information Management tools and IMS Version 13 Compatibility page at this website:

http://www.ibm.com/support/docview.wss?uid=swg21611198

Refer to this page before you start a new IMS migration to access the latest tools. This page includes a table that provides the minimum maintenance that is required for IMS Tools to support IMS 13.

This table features the following keys:

- NS (not supported): The product does not function properly on IMS 13.
- T (tolerate): The product can run as it did on a previous release of IMS, but does not use any of the new IMS V13 functions.
- E (exploit): The product uses the new functions and features that are available in IMS 13.

Note: If an older release of a product is not listed in the matrix, it is not supported on IMS 13. For more information about IMS 13 compatibility for subsequent releases or products that are not included in this matrix, contact your IBM Sales Representative.

12.2.5 Product lifecycle for DB2 and IMS Tools

The IBM DB2 and IMS Tools Product Lifecycle matrix displays the marketing and support services that are available to you over the life of your product. For more information about the product lifecycle, see the Product Lifecycle for DB2 and IMS Tools page at this website:

https://www.ibm.com/support/docview.wss?rs=434&uid=swg27008621

You can select the dates that are listed to go directly to the product announcement letter.

Although every effort is made to provide accurate information, the authoritative source for product information is the IBM announcement letter page, which includes a table with the following keys:

- PID: Product number.
- GA: General Availability. Date with a link to the announcement letter.
- EOM: End of Marketing. Date with a link to the withdrawal letter.
- EOS: End of Support. Date with a link on the service discontinuance announcement.

- TBD: To Be Determined
- Replacement Product: The replacement product name and version if it is no longer supported

> **Note:** For products that are not listed in the Product Lifecycle matrix or for more IBM Software product lifecycle information that is licensed under the International Program License Agreement (IPLA), see IBM Software Support Lifecycle at this website:
>
> `http://www.ibm.com/software/support/lifecycle/`

12.2.6 IBM DB2 and IMS Tools PTF listing

The IMS Tools products are categorized by function. The information describes the minimum version and release levels of the IBM IMS Tools products that support IMS 13.

For more information about the current PTFs for these tools, see IBM DB2 and IMS Tools PTF listing that is available at this website:

`http://www.ibm.com/support/docview.wss?rs=434&context=SSZJXP&uid=swg27008646`

Fix lists are summaries of problem descriptions (APARs) and fixes (PTFs) by product and version. You can use the CSV file to import the list of APARs into a spreadsheet or database for sorting or downloading.

When an APAR closes, a few days might pass before the APAR is available in the table and CSV file.

12.3 IBM Tools Base for z/OS V1.4

IBM Tools Base (previously known as IBM IMS Tools Base for z/OS) contains common infrastructure that is used by IBM DB2 and IMS Tools for z/OS.

The package contains all IBM IMS Tools common infrastructure components that IMS Tools Solution Packs require.

12.3.1 Tools in IBM Tools Base V1.4

IBM Tools Base V1.4 includes the following tools (with product number 5655-V93) that automatically evaluate database health by using data that is provided by IMS Database Solution Pack and IMS Fast Path Solution Pack:

- IBM Tools Base Administration Console
- IBM Tools Base Autonomics Director
- IMS Tools Knowledge Base
- IMS Tools Common Services
- IBM Tools Base Policy Services
- IBM Tools Base Distributed Access Infrastructure

IBM Tools Base V1.4 also includes the IMS Hardware Data Compression Extended component.

These components automatically monitor database health, identify problems, and investigate identified problems via a graphical interface.

12.3.2 IBM Tools Base Administration Console

IBM Tools Base Administration Console for z/OS (Administration Console) is an application server that consolidates key IMS information into a single, intuitive, graphical web interface.

This interface is used on a client to connect to different IMS Tools and IBM Tools Base for z/OS components. Administration Console can accelerate your analysis of your IMS environment and reduce the need for advanced IMS skills.

You can use Administration Console to manage automated database evaluations and monitor IMS environments across an enterprise. You can view aggregate statistics for groups of IMS resources and detailed information for individual IMS resources. You can use this information to identify databases that require your attention and to prevent problems.

The information in Administration Console is gathered from various sources, including IBM Tools Base for z/OS components, other IMS Tools products, and IMS Operations Manager.

Figure 12-1 shows a typical Administration Console environment.

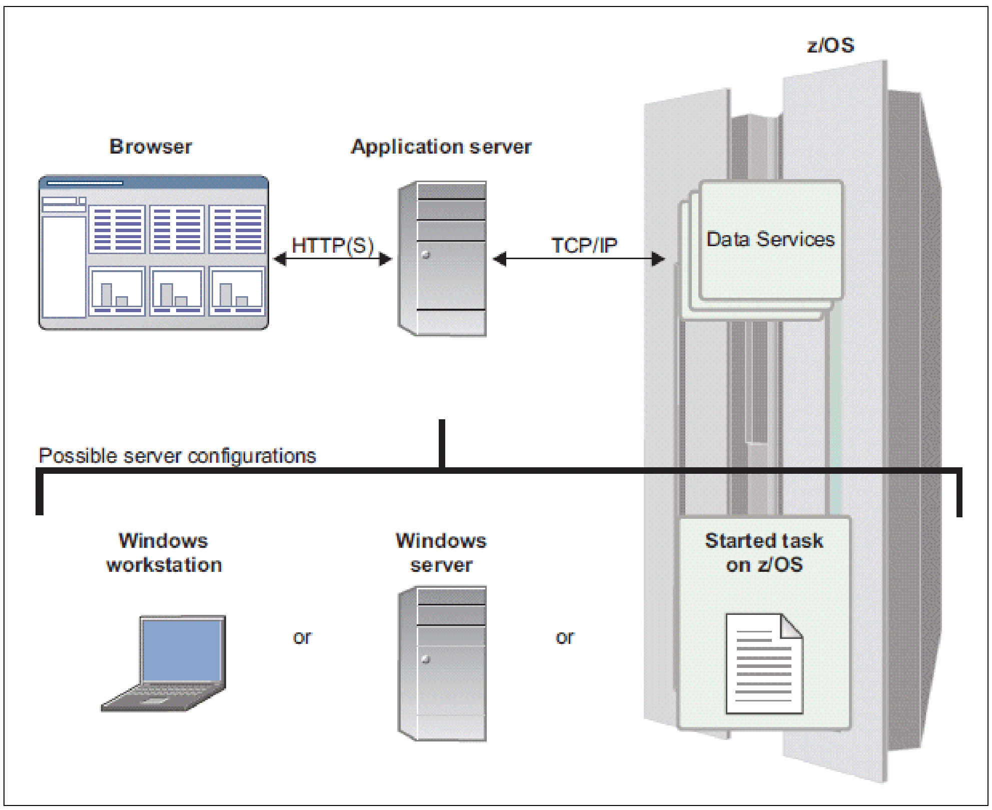


Figure 12-1 Typical Administration Console environment

For more information, see the following resources:

- *IBM Tools Base for z/OS Version 1 Release 4 Administration Console User's Guide*, SC19-3764
- The YouTube IBM IMS channel video, which is available at this website:

  https://www.youtube.com/watch?v=_vvqdqHhAXQ

12.3.3 IBM Tools Base Autonomics Director

IBM Tools Base Autonomics Director for z/OS automates the following recurring database monitoring and maintenance tasks, including automatic and on-demand scheduling of database sensor data collection and evaluation:

- Reorganizations of IMS databases

  Based on policies that are defined by the user, Autonomics Director provides recommendations for the reorganization of databases, partitions, and areas, and automatically starts and manages groups of reorganizations.

- IOVF and SDEP extensions of Fast Path DEDB areas

  Based on policies that are defined by the user, Autonomics Director provides recommendations for extending the independent overflow (IOVF) and sequential dependent (SDEP) portions of Fast Path DEDB areas and automatically starts these processes.

- Automatic collection of sensor data

  Sensor data is collected to capture the status of databases for a specific point.

- Database health evaluations through sensor data content by using Policy Services

  You can define policies and rules in Policy Services, which Autonomics Director uses when it evaluates the most recent database sensor data. The policies are stored in IMS Tools Knowledge Base and are accessed by Autonomics Director indirectly through Policy Services.

- Automatic discovery of databases in your environment

  Autonomics Director provides an automatic database discovery feature that identifies which databases are present in your IMS environment.

- Scheduling data collection and policy evaluation

  You can define parameters that control how frequently data is collected and how frequently policies are evaluated by Autonomics Director.

Autonomics Director server

The Autonomics Director server is responsible for the control and flow of the Autonomics Director environment.

Autonomics Director has three types of servers: a master server, and active and failover servers.

Figure 12-2 on page 374 shows the master, active, and failover servers in an Autonomics Director environment.

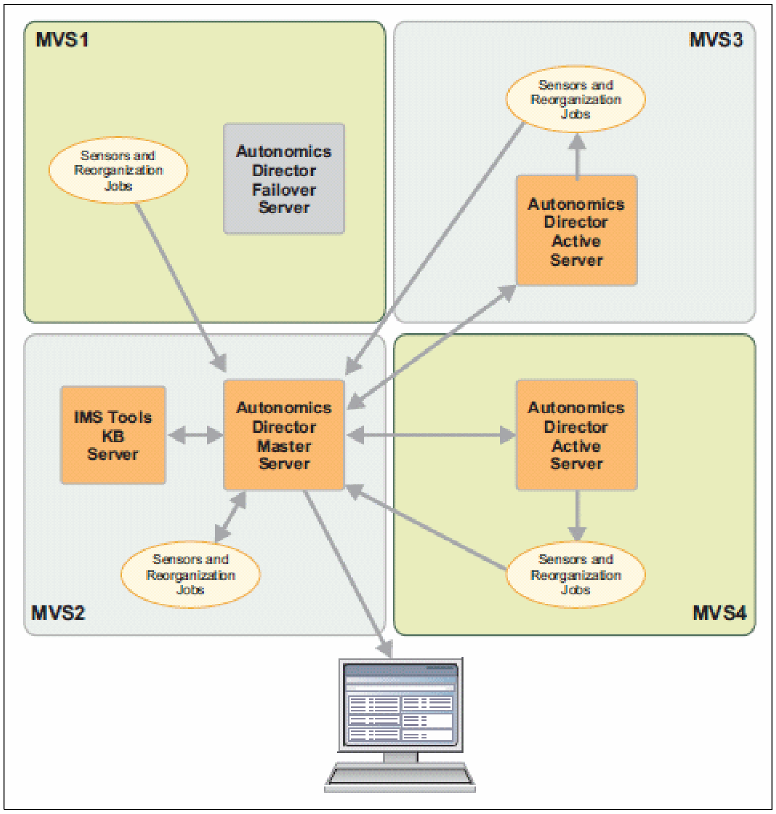


Figure 12-2 Autonomics Director server environment

For more information, see *IBM Tools Base for z/OS Version 1 Release 4 Autonomics Director User's Guide*, SC19-3765.

12.3.4 IMS Tools Knowledge Base

IMS Tools Knowledge Base (ITKB) provides a common Information Management service that allows the sharing of data that is generated by multiple tool products within a sysplex from a single, centralized interface.

The following types of information can be provided:

- Reports from IMS Tools or third parties.
- Sensor data: Information that is captured by an IMS Tools product (such as IMS Reorganization Expert 410) at an instance in time that represents the condition, or state, of one or more databases.
- Policies: Sets of rules that define threshold limits for specific types of database conditions. These threshold limits are then evaluated against your sensor data.

This data is centralized in a common report repository and can be managed with its viewing interface through a complex sysplex environment. Historical report data can be found through a powerful search, and preserved for future decision making.

ITKB is the single interface within a sysplex environment for multiple IMS Tools products to share report output.

By using the ITKB report service, automatic capturing of reports are generated by participating IMS Tools products and storing these reports in a central report repository.

Figure 12-3 shows the interaction of these components within a sysplex.

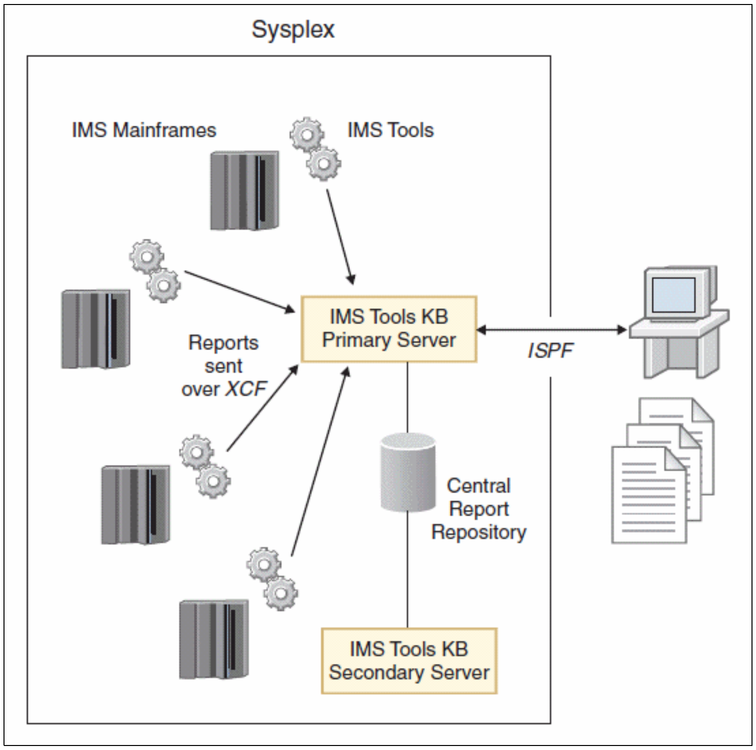


Figure 12-3 Tools Base IMS Tools Knowledge Base report service environment

For more information, see *IBM Tools Base for z/OS Version 1 Release 4 IMS Tools Knowledge Base User's Guide*, SC19-3768.

12.3.5 IMS Tools Common Services

IMS Tools Common services include the IMS Tools Generic Exits and the IMS Tools Online System Interface (TOSI) functions.

IMS Tools Generic Exits

IMS Tools Generic Exits is a set of specialized utilities that are used by IMS Tools product code to manage various interfaces, including the managing of IMS exits.

They are a collection of exit routines that can call multiple exit routines from a single exit point in an IMS environment.

IMS Tools Online System Interface

IMS Tools Online System Interface (TOSI) is a general-purpose command interface that allows IMS Tools to interface with all supported IMS versions.

TOSI is a component of the IBM Tools Base for z/OS and is a prerequisite for multiple IMS Tools. It can also be shared with multiple IMS Tools.

For more information, see *IBM Tools Base for z/OS Version 1 Release 4 IMS Tools Common Services User's Guide*, SC19-3767.

12.3.6 IBM Tools Base Policy Services

IBM Tools Base Policy Services for z/OS (also referred to as Policy Services) is a core IMS Tools technology that supports conditional autonomic database health management functionality for participating IMS Tools products.

Policy Services can monitor specific database state by evaluating the sensor data that is collected by an IMS Tools product, and by providing a response to any conditions that exceed the threshold values specified for this state.

Conditional autonomics can provide the following functionality:

- Evaluate whether there is a need for any specific database maintenance operation to occur.
- Make recommendations for corrective actions that are based on user-defined (policy-driven) requirements.

Figure 12-4 shows a conditional reorganization scenario.

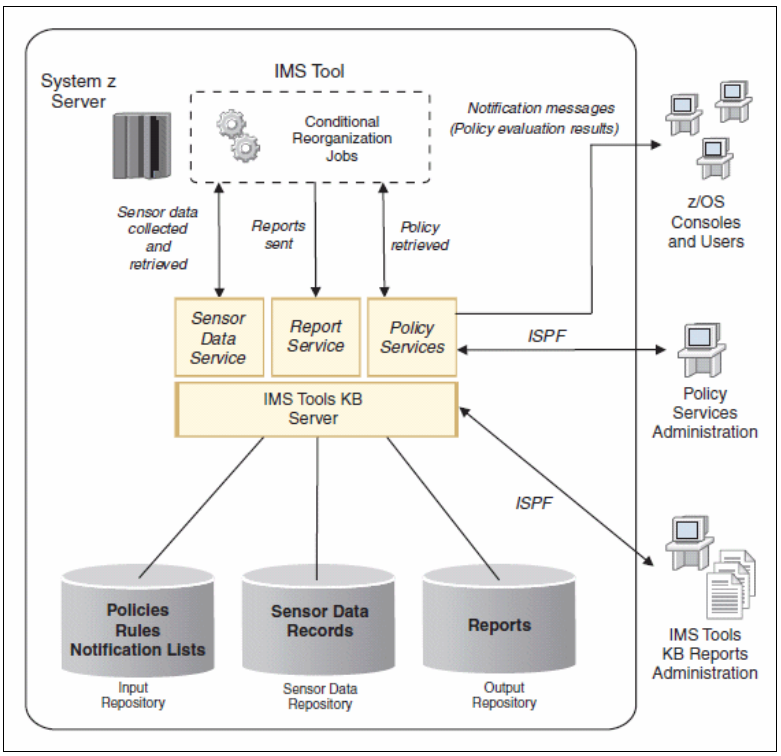


Figure 12-4 Example conditional reorganization scenario

For more information, see *IBM Tools Base for z/OS Version 1 Release 4 Policy Services User's Guide*, SC19-3770.

12.3.7 IBM Tools Base Distributed Access Infrastructure

IBM Tools Base Distributed Access Infrastructure for z/OS (which is also referred to as Distributed Access Infrastructure) enables authorized access to configured IMS Tools from authenticated TCP/IP clients. It acts as a gateway for communication between distributed platforms and z/OS and is delivered as a component of the Tools Base.

The Distributed Access Infrastructure includes the following features:

- It is a set of software components that enable distributed clients access to IMS Tools through standard TCP/IP socket communication.
- It acts as a key component in extending the availability of IMS Tools to workstation and browser-based interfaces.

Distributed access operation

Remote clients must first establish an authenticated connection to the Distributed Access Infrastructure by passing a user ID and password. After authentication, the client program can send and receive messages to and from the IMS Tools that the clients are authorized to access.

Distributed Access Infrastructure operates in a z/OS environment and uses a TCP Server, the Tools Access Server (TAS), and the Subordinate TAS (SOT) to provide distributed access to IMS Tools products from remote systems.

Figure 12-5 shows the overall flow of communication from the distributed systems to the IMS Tools products.

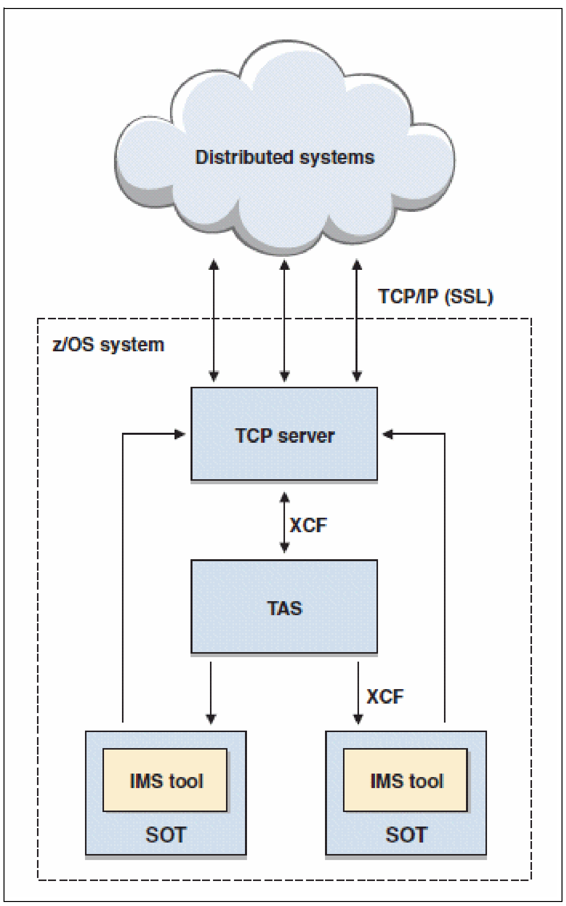


Figure 12-5 Distributed Access Infrastructure architecture

For more information, see *IBM Tools Base for z/OS Version 1 Release 4 Distributed Access Infrastructure User's Guide*, SC19-3771.

12.3.8 IMS Hardware Data Compression Extended

IMS Hardware Data (HD) Compression Extended provides functions for compressing IMS data. IMS HD Compression Extended uses the z/OS hardware data compression (HDC) that is available on IBM processors.

IMS provides a basic utility that supports z/OS HDC. IMS HD Compression Extended provides enhanced functions that support z/OS HDC that are not available in the basic IMS utility.

IMS HD Compression Extended can be used to compress HDAM, HIDAM, HISAM, and Fast Path databases.

IMS HD Compression Extended benefits

Compression features the following advantages:

- Reduced DASD storage requirements for databases
- Reduced database I/O
- CPU usage is not significantly increased

Compressing data conserves DASD and tape resources. It also reduces high RBA usage, which is especially beneficial to IMS databases that are approaching the 4 GB limit for VSAM or 8 GB limit for OSAM.

Compression also can improve database response time and performance because fewer physical I/Os are needed to access data.

How IMS HD Compression Extended works

IMS HD Compression Extended provides a set of tools that are designed to help implement and manage compressed IMS databases. These tools feature the following advantages:

- An easy-to-use approach for creating and maintaining compressed IMS databases.
- Features to evaluate compression impact, build, verify, and archive dictionaries, implement compression, and monitor and maintain production compression routines.
- Can use generic, enterprise, or custom compression dictionaries.
- An ISPF interface that is used easily.

Segment data is expanded and compressed by the compression exit routine whenever a segment is accessed.

The compression routine contains two dictionaries: one for compression and one for expansion (collectively referred to as the dictionary).

For more information, see *IBM Tools Base for z/OS Version 1 Release 4 IMS Hardware Data Compression Extended User's Guide*, SC19-3769.

12.4 IMS Fast Path Solution Pack for z/OS V1.3

IMS Fast Path Solution Pack for z/OS V1.3 applies to data entry databases (DEDBs).

It provides features and functions to implement preferred practices for managing an IMS Fast Path environment.

It can help boost system availability by enabling the operation of various key functions without taking the IMS databases offline.

12.4.1 Tools in IMS Fast Path Solution Pack V1.3 (5655-W14)

IMS Fast Path Solution Pack V1.3 includes the following tools:

- IMS High Performance Fast Path Utilities:
  - IMS Fast Path Advanced Tool
  - IMS Fast Path Online Tools
  - IMS Fast Path Basic Tools
  - Database Sensor
- IMS High Performance Image Copy for z/OS V4.2 (5655-N45)
- IMS Library Integrity Utilities for z/OS V2.2 (5655-U08)
- IMS Database Repair Facility

12.4.2 Highlights

IMS Fast Path Solution Pack provides the following complete tool set:

- Online/Offline
- Reorganization and restructure
- Pointer check and space analysis
- High performance/function Image Copy
- Library consistency verification

It delivers performance by optimization of resource and CPU usages with reduced elapsed time and the new Fast Path Secondary Index support.

IMS Fast Path Solution Pack also provides Advanced DB Monitoring Solution, which features the following benefits:

- Simplification through automation by Autonomics Director
- Modernization through graphical interface by Administration Console

12.4.3 Installing IMS Fast Path Solution Pack

The installation includes the following requirements:

- z/OS V1.12 or higher
- Tools Base for z/OS, V1.4

The supported IMS versions are IMS 11, IMS 12, and IMS 13.

Before you install IMS Fast Path Solution Pack, you must complete the SMP/E installation for Tools Base. Consider the following points:

- Except for the base product IMS High Performance Fast Path Utilities, the other products in the Pack can be installed only if necessary.
- It is not necessary any more to install IMS Fast Path Solution Pack and Tools Base in the same CSI.

GA PTF (APAR PI05155) must be applied after you install IMS Fast Path Solution Pack so that new functions and features are included.

For more information about the checklist to install and customize IMS Fast Path Solution Pack, see *IMS Fast Path Solution Pack for z/OS V1.3: Overview and Customization*, SC19-4009.

12.4.4 Fast Path Solution Pack evolution

Figure 12-6 shows the Fast Path Solution Pack evolution.

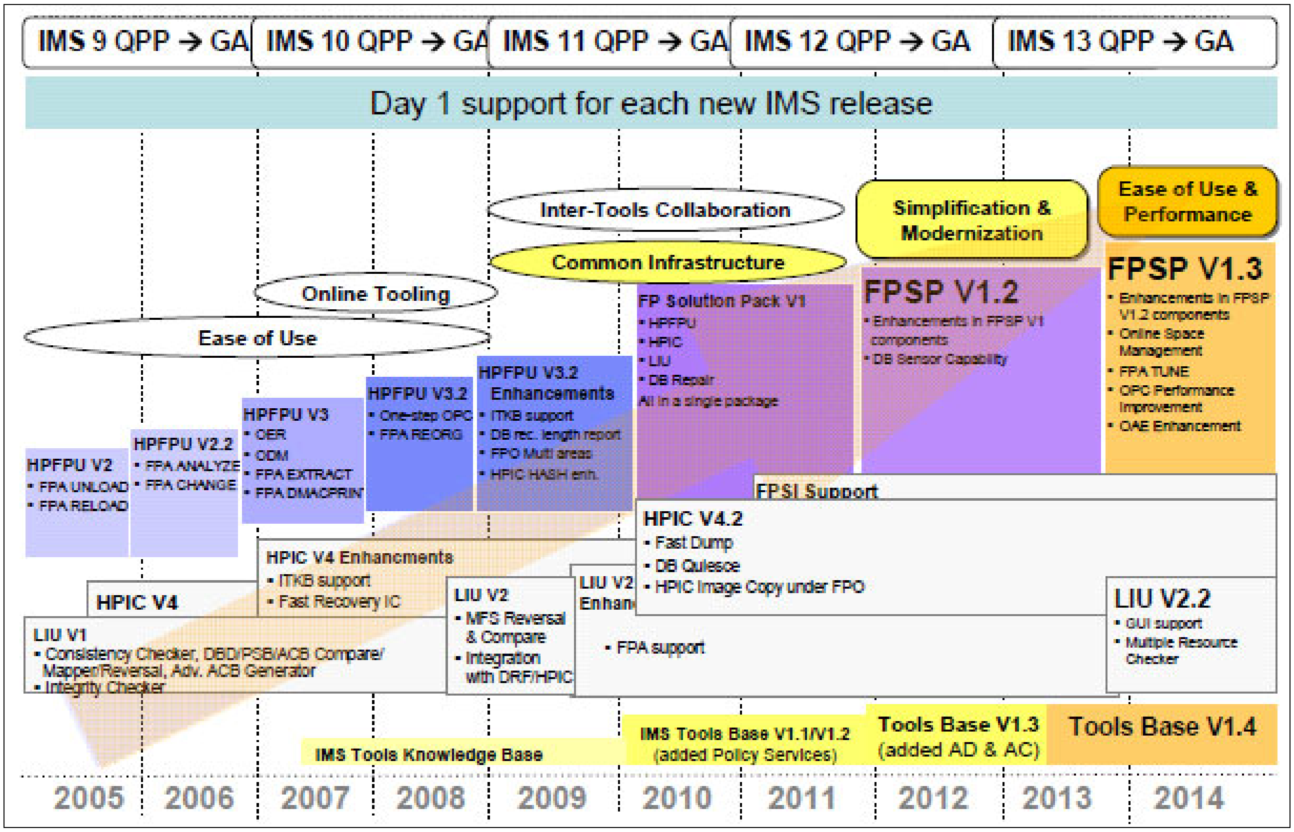


Figure 12-6 Fast Path Solution Pack evolution

Figure 12-7 shows an example of using Fast Path Solution Pack.

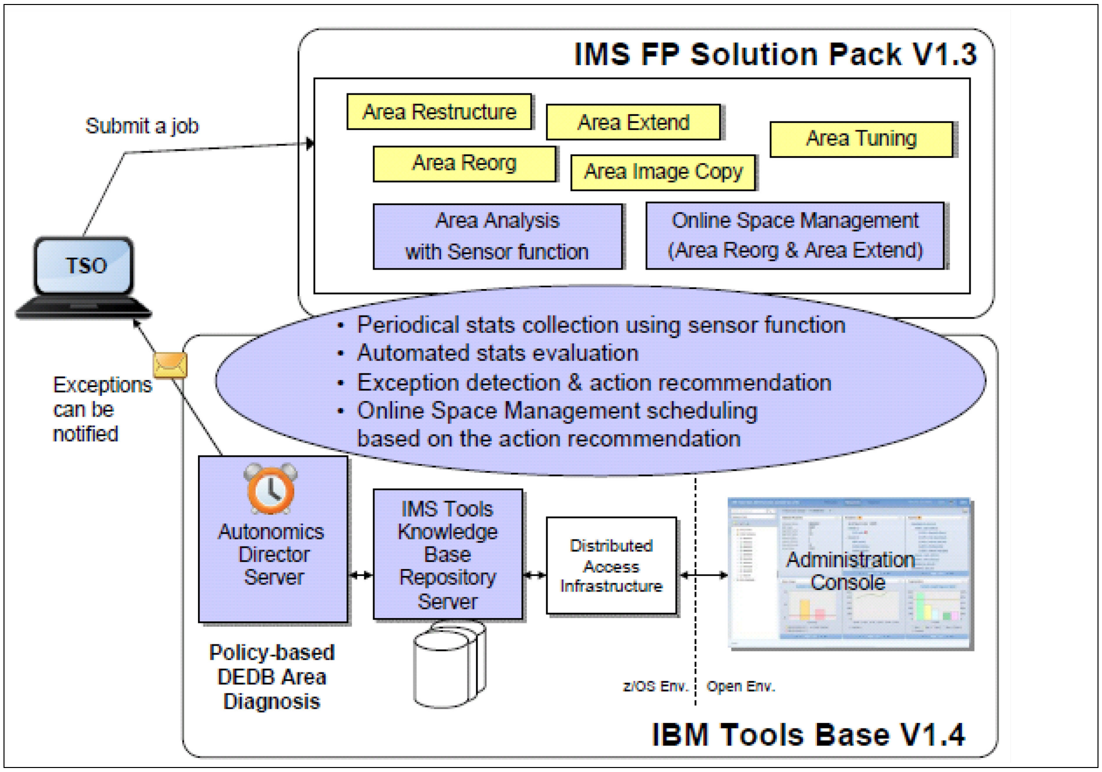


Figure 12-7 Example for using Fast Path Solution Pack

12.4.5 DEDB utility solutions

Figure 12-8 on page 382 shows the DEDB utility solutions.

Category	Function	FPA (offline)	FPB (offline)	FPO (online)	OSM* (online) New
Reorg	Unload	UNLOAD	DEDB Unload	Online Data Extract (ODE)	
	Reload	RELOAD	DEDB Reload		
	Change	CHANGE			
	Reorganize	REORG		Online Expert Reorg (OER)	OER capability
	Build index	INDEXBLD			
	Extend			Online Area Extender (OAE) Enhanced	OAE capability
Analyze	Integrity verification and analysis	ANALYZE	DEDB Pointer Checker	Online Pointer Checker (OPC) Enhanced	OPC capability
	Print DMAC	DMACPRT		Online DMAC Print (ODM)	
Extract	Extract segments	EXTRACT	Unload/ Reload user exit routine	Online Data Extract (ODE)	
Other	Tuning Aid	TUNE New	DEDB Tuning Aid	Combination of OPC and TUNE New	
	Initialize	INITIALIZE			

* OSM: Online Space Management utility

Figure 12-8 DEDB utility solutions

12.4.6 IMS High Performance Fast Path Utilities

IMS High Performance Fast Path Utilities (also referred to as IMS HP Fast Path Utilities) provides extensive utilities that include all of the functions that you need to analyze, maintain, and tune IMS Data Entry Databases (DEDBs).

They also provide features that can help increase system availability by enabling the operation of various key functions without taking your IMS databases offline.

IMS HP Fast Path Utilities includes the following main tools:

- IMS Fast Path Advanced Tool (FPA)
- IMS Fast Path Basic Tools (FPB)
- IMS Fast Path Online Tools (FPO)
- Online Space Management utility (OSM)

In addition to these main tools, IMS HP Fast Path Utilities includes supplementary utilities.

Key functional changes of IMS HP FP Utilities

The Tuning Aid function was added to Fast Path Advanced Tool and Fast Path Online Tools to simulate changes in the DBD parameters and randomizers without loading the segments.

A user can select the physical database attributes that satisfy the performance and space utilization requirements.

Elapsed time of Online Pointer Checker (OPC) with FULLSTEP=YES is reduced by processing multiple areas concurrently and eliminating I/Os for intermediate data sets. CPU time of OPC with FULLSTEP=YES is also reduced.

Online Area Extender (OAE) was enhanced for ease of use with simpler parameter specification. Consider the following points:

- The extended size for SDEP portion can be specified with a specific number or by a percentage of CIs within the total number of CIs that is defined in the SDEP portion.
- The extended size for IOVF portion can be specified with a specific number of by a percentage of UOWs within the total number of UOWs that is defined in the IOVF portion.
- These specifications allow OAE to be run multiple times without JCL modification.

For more information, see *IBM IMS Fast Path Solution Pack for z/OS Version 1 Release 3 IMS High Performance Fast Path Utilities User's Guide*, SC19-4010.

12.4.7 IMS High Performance Image Copy for z/OS V4.2

IBM IMS High Performance Image Copy for z/OS (also referred to as IMS HP Image Copy) creates image copies of a database and recovers a physically damaged data set of an IMS database. The Recovery function is an enhanced base recovery utility that supports the use of HP IC proprietary IC formats and HP IC compressed ICs.

It helps you speed database backup time by supporting quick shots of image copies and restarting methods. As a result, you can help users become more productive and avoid expensive losses from missed business opportunities.

IMS HP Image Copy reads an image copy and creates one or more copies while allowing a change in IC compression. It not only provides the same functions as the IMS standard utilities, which are the Database Image Copy utility (DFSUDMP0) and the Database Recovery utility (DFSURDB0), but provides certain unique functions, such as to create various types of image copies, generate a copy data set from the original image copy data set, and set site default values for IMS HP Image Copy jobs.

The runtime interface of IMS HP Image Copy is a command-like interface, which provides ease of use. It can replace the IMS standard utilities.

IMS HP Image Copy functions

IMS HP Image Copy creates as-is image copies of a database data set. You can create the following copies:

- Batch image copy

  Batch image copies are created while no other applications are actively updating the database. Therefore, a batch image copy is a clean copy of the database.
- Concurrent image copy (CIC)

  Concurrent image copies can be created even when IMS or IMS applications are updating the database. Therefore, a concurrent image copy is a fuzzy copy of the database.

IMS HP Image Copy also provides the following features:

- Recovery function

  Recovers a physically damaged data set by using an image copy data set.
- Create Image Copy function

  Reads an image copy data set, and creates one or more copies of it.

Features and benefits

IMS HP Image Copy provides the following significant features that simplify your database backup and recovery tasks:

- Reduces copy and recovery time.
- Provides a powerful engine and parallelism capabilities to reduce CPU and elapsed time.
- Allows higher database availability by passing commands to IMS.
- Tightly integrates with other tools to give better performance and simplify the copy process.

Figure 12-9 shows the interactions with the IMS Tools and the components of IBM Tools Base.

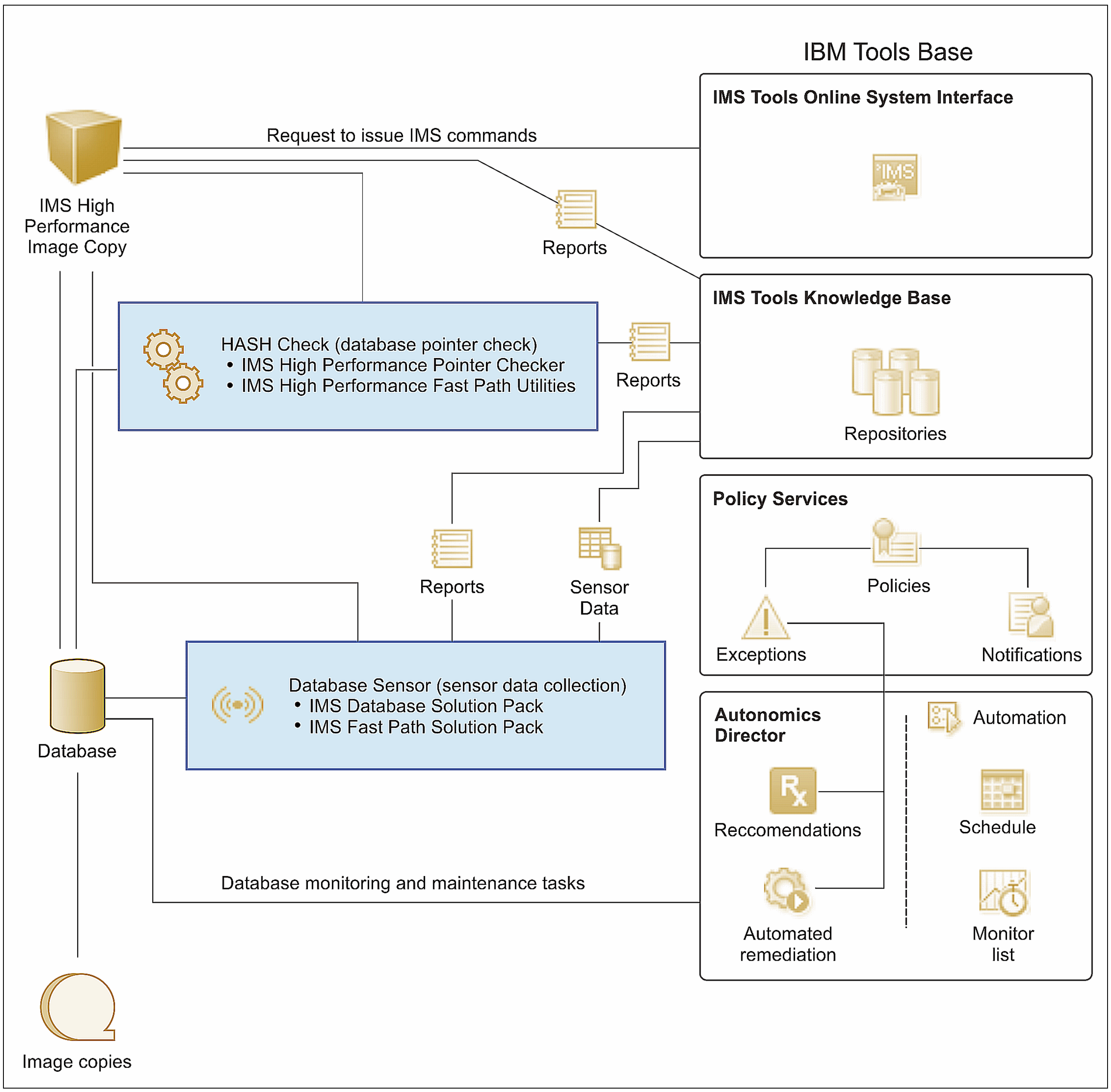


Figure 12-9 Interactions of IMS HP Image Copy with other IMS tools and the components of IBM Tools Base

For more information, see *IBM IMS High Performance Image Copy for z/OS Version 4 Release 2 User's Guide*, SC19-2756.

12.4.8 IMS Library Integrity Utilities for z/OS V2.2

IBM IMS Library Integrity Utilities for z/OS (also referred to as IMS Library Integrity Utilities or IMS LIU) aids you in managing data for the libraries, such as the DBD libraries, PSB libraries, ACB libraries, and RECON data sets that you use when referring to IMS databases.

The following functions are typical of data management:

- Prevent the database corruption that the use of an incorrect member of a library can cause.
- Check the consistency among each library.
- Check, compare, change, generate, and maintain the members of a library.

These functions are provided by the utilities that are described in this section.

Integrity Checker utility

This utility prevents DB corruption that is caused by using an incorrect DBD. An incorrect DBD is a DBD that is different from the one used to load the database.

This situation can arise if you use an old DBD after a DBD change is applied.

Consistency Checker utility

This batch utility ensures that the following definitions in IMS were created for your database:

- An ACB in an ACB library
- A Database Definition entry in the MODBLKS module (DFSDDIRx)
- A DFSMDA dynamic allocation member for database data sets in an MDA library
- DB and DSG registration in recovery control (RECON)

Multiple Resource Checker utility

For DBDs, Multiple Resource Checker (MRC) verifies whether the following resources exist in each library and whether the contained definitions are consistent:

- DBD members in DBD libraries
- ACB members in ACB libraries
- DB and DBDS records registered in RECON data sets

For PSBs, Multiple Resource Checker verifies whether the following definitions are the same in each library:

- PSB members in PSB libraries
- ACB members in ACB libraries

Multiple Resource Checker compares all of the RECON, DBD, PSB, and ACB libraries across multiple IMS environments in one job step. The job can perform the following tasks:

- Process different IMS versions
- Present the result that shows consistency at a glance
- Optional report shows the detailed differences of RECONs

DBD/PSB/ACB Compare utility

This batch utility reports the difference between DBD, program specification block (PSB), and application control block (ACB) members that have the same name but are in separate object libraries, whatever their type is (same or different).

DBD/PSB/ACB Mapper utility

This batch utility produces a printed map (picture of segment hierarchy) from DBDs, PSBs, and ACBs.

DBD Map Viewer utility

Supports the DBD Map Viewer, which is an add-on feature to Tools Base Administration Console to view the graphical visualization of a database structure map, the DBD macro source, and the DBD XML document. You can perform the following tasks:

- Easily view the binary DBD data that is stored in IMS DBD libraries.
- Display the DBD information in an XML format that conforms to the format of the XML instance document from IMS Catalog.
- Save and export DBD maps, DBD macro statements, DBD XML documents, and segment lists to a local drive for offline analysis.

For more information, see IMS Library Integrity Utilities DBD Viewer at the following YouTube IBM IMS channel:

https://www.youtube.com/watch?v=rUqgI2WHs4M

DBD/PSB/ACB Reversal utility

This batch utility converts a DBD, PSB, or ACB member back into IMS **DBDGEN** or **PSBGEN** utility control statements.

Advanced ACB Generator utility

This batch utility provides a high-speed generation process for processing large volumes of IMS ACBs. It replaces the IMS **ACBGEN** utility.

ACBLIB Analyzer utility

This utility verifies that all ACB library members are at the same IMS version or release level, and that all members were placed in the ACB library by the ACBGEN process; that is, the library was not inadvertently used during a DBDGEN or PSBGEN.

MFS Reversal utility

This batch utility can compare two MFS load module libraries and convert the format in MSF source format.

MFS Compare utility

MFS Compare is designed to compare two sets of MFS format control blocks from two MFS format libraries to quickly highlight the differences between them.

Figure 12-10 on page 387 shows how Integrity Checker prevents database corruptions for IMS online applications. If a mismatch is found between the two DMBs, the DMB verification process modifies the response from DBRC to deny the DBRC authorization request.

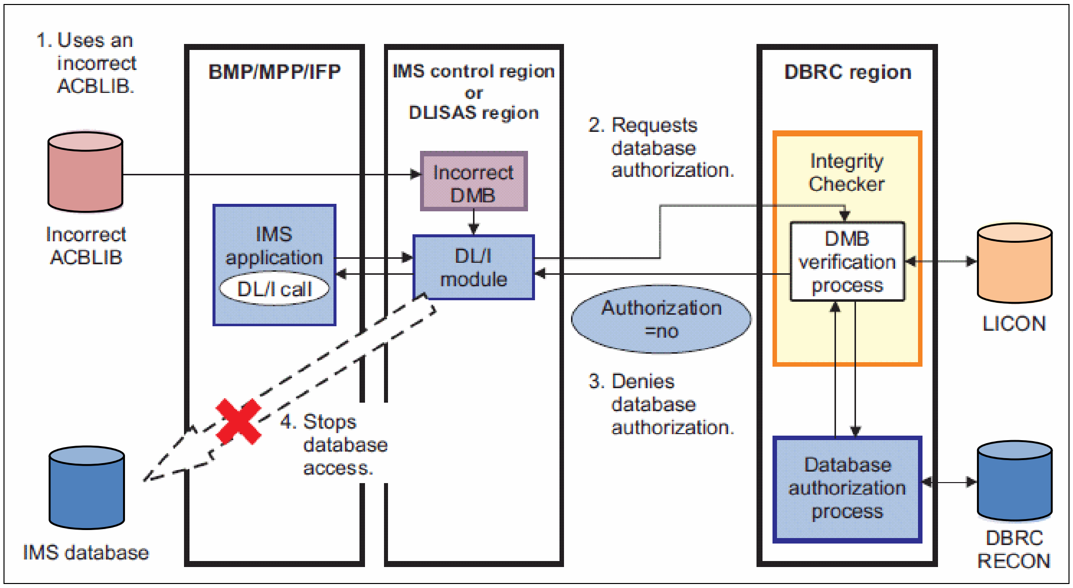


Figure 12-10 How Integrity Checker prevents database corruption in IMS online applications

Catalog Manager

The new Catalog Manager utility was added by the Library Integrity Utilities by APAR PI21200. The utility helps users who are implementing the IMS Catalog. For more information about this utility, see the following resources:

- http://www.ibm.com/support/docview.wss?uid=swg27041486
- *IBM IMS Library Integrity Utilities for z/OS Version 2 Release 2 User's Guide*, SC19-3979
- *IBM IMS Database Solution Pack for z/OS Version 2 Release 1 Overview and Customization*, SC19-4007

12.4.9 IMS Database Repair Facility

IMS Database Repair Facility for IMS Solution Packs (which is also referred to as IMS Database Repair Facility) is a powerful tool to repair VSAM and OSAM organized IMS databases that contain pointer or data errors.

It can be run interactively for VSAM and OSAM data sets or in batch mode that are IMS and non IMS data sets.

Pointer and data corruption occur during power failure situations. They also might result from severe application errors or other mishaps to data sets that are on DASD devices, such as I/O errors.

Pointer errors are found by examining the output of Pointer Checker jobs. Data errors are generally reported by users.

Using IMS Database Repair Facility

After you determine the bad pointer or data, use the interactive IMS Database Repair Facility to browse around the database for other validation purposes. This review is done to verify that other pointers or data were not corrupted.

By using IMS Database Repair Facility, you can change the bad pointers and data to the values you think they should include. Although you use the utmost caution, errors can occur.

To address this problem, when IMS Database Repair Facility is run interactively, it has an Undo feature that reverses pointer and data changes. When IMS Database Repair Facility is used to update a database, the tool keeps a file of before and after pointer and data images.

This file is kept until the next time IMS Database Repair Facility is started and used to create a change against the same database.

Use the Pointer Checker utility immediately after you update your database. Then, if an error is detected, you can use the interactive IMS Database Repair Facility's Undo function to back out the change.

This information is also true for the batch operation of IMS Database Repair Facility. However, no Undo facility is available. Instead, you can reverse changes that are made in error by using information that is provided in the Control Statements and Messages report.

For more information, see *IBM IMS Database Repair Facility for IMS Solution Packs User's Guide*, SC19-2916.

12.5 IMS Database Solution Pack for z/OS V2.1

IMS Database Solution Pack for z/OS V2.1 (DB Solution Pack) provides a comprehensive set of high performance utilities to unload, load, index build, reorganize, backup, verify, and report on full-function databases while helping to reduce the operational complexity and the impact of database reorganization on system resources.

It integrates the entire set of IMS database products to help you to manage IMS full function and high availability large databases (HALDB) with a single consolidated solution.

IMS Database Solution Pack is specifically designed to provide database administrators (DBAs) with the following smart solutions to database operation and maintenance tasks:

- Database monitoring
- Database reorganization
- Database optimization

12.5.1 Installing IMS Database Solution Pack

The IMS Database Solution Pack included the following installation requirements:

- z/OS V1.12 or higher
- IMS versions 11, 12, or 13

12.5.2 Tools in IMS Database Solution Pack V2.1

The following tools and their product numbers are available in IMS Database Solution Pack V2.1:

- IMS Database Reorganization Expert V4.1 (5655-S35)
- IMS High Performance Unload V1.2 (5655-E06)
- IMS High Performance Load V2.1 (5655-M26)
- IMS High Performance Prefix Resolution V3.1 (5655-M27)
- IMS Index Builder V3.1 (5655-R01)
- IMS High Performance Image Copy V4.2 (5655-N45)
- IMS High Performance Pointer Checker V3.1 (5655-U09)
- IMS Database Repair Facility

- IMS Library Integrity Utilities V2.2 (5655-U08)
- IMS High Availability Large Database Toolkit V2.1
- IMS Online Reorganization Facility V2.1
- Database Sensor (available when Database Reorganization Expert is installed)
- DBD Map Viewer (an LIU extension for ITB V1.4 Administration Console)

12.5.3 Highlights

IMS Database Solution Pack provides a complete tool set for the following database administration functions:

- Conditional reorganization
- Online, Read-Only, and offline reorganization
- Unload and Reload User Exits
- Unload and Reload application APIs
- Pointer check and space analysis
- High performance function Image Copy
- Library consistency verification
- Specialized HALDB Support

It delivers performance by optimization of resource and CPU usages with reduced elapsed time.

IMS Database Solution Pack also provides Advanced DB Monitoring Solution, which includes the following features:

- Simplification through automation by Autonomics Director
- Modernization through graphical interface by Administration Console

12.5.4 IMS Database Reorganization Expert V4.1

Database reorganization typically involves complex analysis tasks. Generally, these time-consuming tasks require knowledge, expertise, and experience in space management of IMS databases.

IMS Database Reorganization Expert (RGE) facilitates the database reorganization tasks for IMS full-function databases by helping you automate and eliminate the complex tasks that pertain to database reorganization.

The main utility of IMS Database Reorganization Expert, Smart Reorg utility, offers two significant features:

- Conditional Reorganization Support Service (also referred to as CRSS)
- Parallel Reorganization Service

Conditional Reorganization Support Service

The Conditional Reorganization Support Service provides the database diagnosis function and the database status reporting function. It helps you by perform the following tasks:

- Evaluating an IMS full-function database and determining the need for reorganization
- Requesting the reorganization process only when a database must be reorganized (based on the results of database evaluation)
- Reevaluating the reorganized database to check the effect of the reorganization
- Providing a comprehensive summary report on the database status and, when the database is reorganized, the change in the status

Parallel Reorganization Service

The Parallel Reorganization Service provides a high-performance reorganization function. It helps you by performing the following tasks:

- Running the unload and the reload functions concurrently in a single job step
- Running the optional functions that typically pertain to reorganization tasks (such as index building, image copying and hash pointer checking, and prefix resolution and update tasks) concurrently in the same job step

The Parallel Reorganization Service provides the reorganization function by internally running the individual high-performance IMS Tools products. A typical environment for running a conditional reorganization job requires the following components:

- IMS Database Reorganization Expert (this product)
- IMS HP Unload
- IMS HP Load
- IMS Tools Knowledge Base (included in Tools Base)
- Policy Services (included in Tools Base)

Sensor data and data elements

Sensor data is the information that is collected by the DB Sensor component for measuring the state of a specific database condition.

The sensor data is handled by the IMS Tools KB server and stored in a central IMS Tools KB Sensor Data repository as records made up of data element values.

The *data element values* that are collected by DB Sensor are used for policy evaluation and are printed in Diagnosis reports.

These data element values help you understand the condition of the database and, if necessary, how to address database exceptions.

IMS Reorganization Expert enhancements

The pre-reorganization utility and post-reorganization utilities are new utilities of IMS Database Reorganization Expert. By using the utilities, the Smart Reorg Utility can reorganize the external logically related databases.

Supporting database types are HDAM and HIDAM.

The function was made available by Reorganization Expert 4.1 APAR PM68661 (PTF UK80515).

The Pre-reorganization utility performs the following pre-reorganization processing that is required for the Smart Reorg utility to reorganize external logical related databases:

- Detection of to-be-processed databases in a logical group
- Automated DBRC authorization
- Automated IMS command

Databases must be registered to DBRC.

The Post-reorganization utility performs the following post-reorganization processing after external logical related databases processed by Smart Reorg Utility.

- Automated DBRC authorization and unauthorization
- Automated IMS command
- Automated data set name swapping
- Dynamic allocation of database data sets

- Prefix resolution and update
- Image copy processing and HASH pointer checking
- Automated DBRC notification
- Databases must be registered to DBRC

You can use Group-Managed Reorganization of IMS Tools Base V1.4 Autonomics Director with the Post-reorganization utility.

Figure 12-11 shows the process flow for reorganizing databases that include external logical relationships.

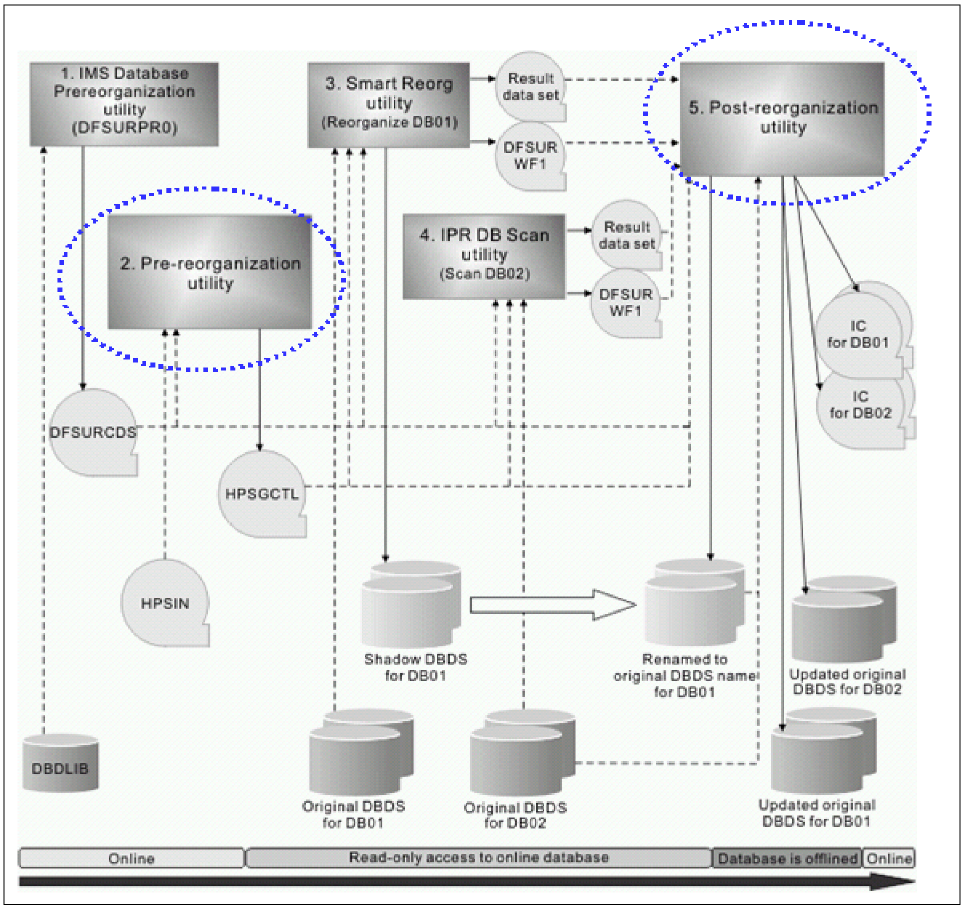


Figure 12-11 Flow for reorganizing databases that include external logical relationships

Building PSINDEXes with HALDB reorganization

Build Partitioned Secondary Indexes (PSINDEXes) with the indexed HALDB reorganization in one job step. This function was made available by APAR PM94789.

The following new control statements are provided:

- 'PSINDEXBLD=YES' to build PSINDEXes
- 'ILDSBLD=YES,INITONLY' to skip loading ILE records to ILDS

These statements are effective when PSINDEXs are rebuilt after HALDB is reorganized in which no logical relationships are defined.

Index Builder and HP Load enhancements

You can now build PSINDEXes without a DB Scan. A new intermediate work file (WFP) is used instead of Index Builder DB Scan. The function helps reduce the total elapsed time of HP Load and Index Builder job steps. This function was made available by APARs PM62216 and PM64823. Figure 12-12 shows the Index builder and HP Load enhancement.

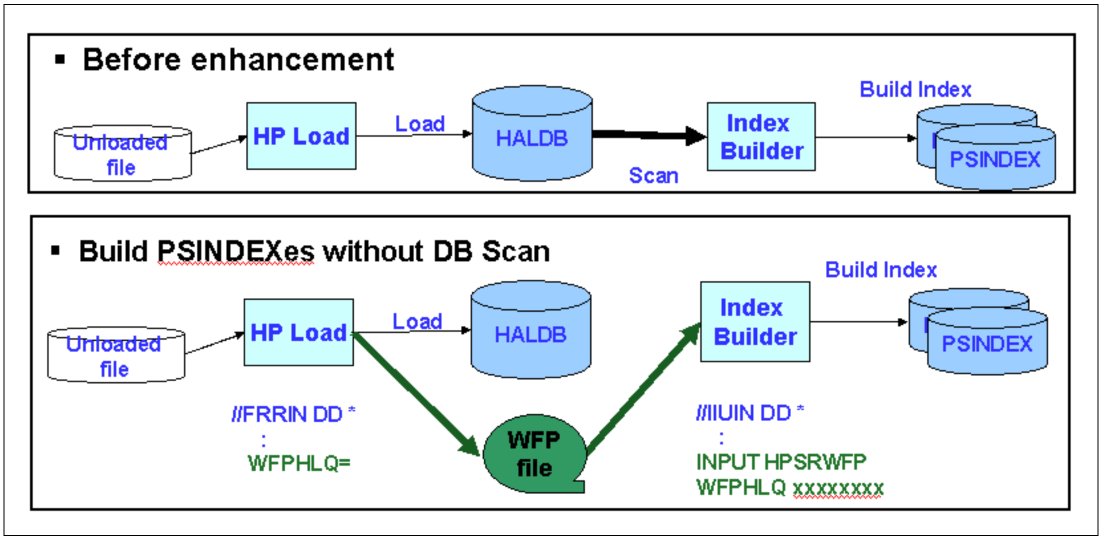


Figure 12-12 Index builder and HP Load enhancement

For more information, see the following publications:

- *IBM IMS Database Reorganization Expert for z/OS Version 4 Release 1 User's Guide*, SC19-1137
- *IBM IMS High Performance Load for z/OS Version 2 Release 1 User's Guide*, SC18-9222
- *IBM IMS Index Builder for z/OS Version 3 Release 1 User's Guide*, SC18-9101

12.5.5 IMS High Performance Unload V1.2

IBM IMS High Performance Unload for z/OS (which is also referred to as IMS High Performance Unload or IMS HP Unload) provides high-speed database unloading and capabilities that are not included in the basic utilities that are provided by IMS.

It improves the performance of IMS data retrieval application programs by using the unload application programming interface (API).

IMS High Performance Unload components

IMS High Performance Unload includes two unload utilities, FABHURG1 and FABHFSU that provide high-speed unloading capability. It also includes an application programming interface (API) for DL/I application programs that use GN calls.

IMS HP Unload is designed for use with IMS Database Reorganization Expert for z/OS and IMS Online Reorganization Facility for z/OS, and with other high-performance IMS Tools products to provide the most efficient solution for IMS database reorganization.

IMS HP Unload replaces the functionality of the IMS HD Reorganization Unload utility (DFSURGU0).

IMS HP Unload is serviced by a high-performance database retrieval engine called High Speed Sequential Retrieval (HSSR) Engine. IMS HP Unload also has an API that is compatible with the HSSR call API that is provided by High-Speed Sequential Retrieval of IMS System Utilities, Data Base Tools (PID: 5685-093).

Figure 12-13 shows an overview of the structure of IMS High Performance Unload and its data flow.

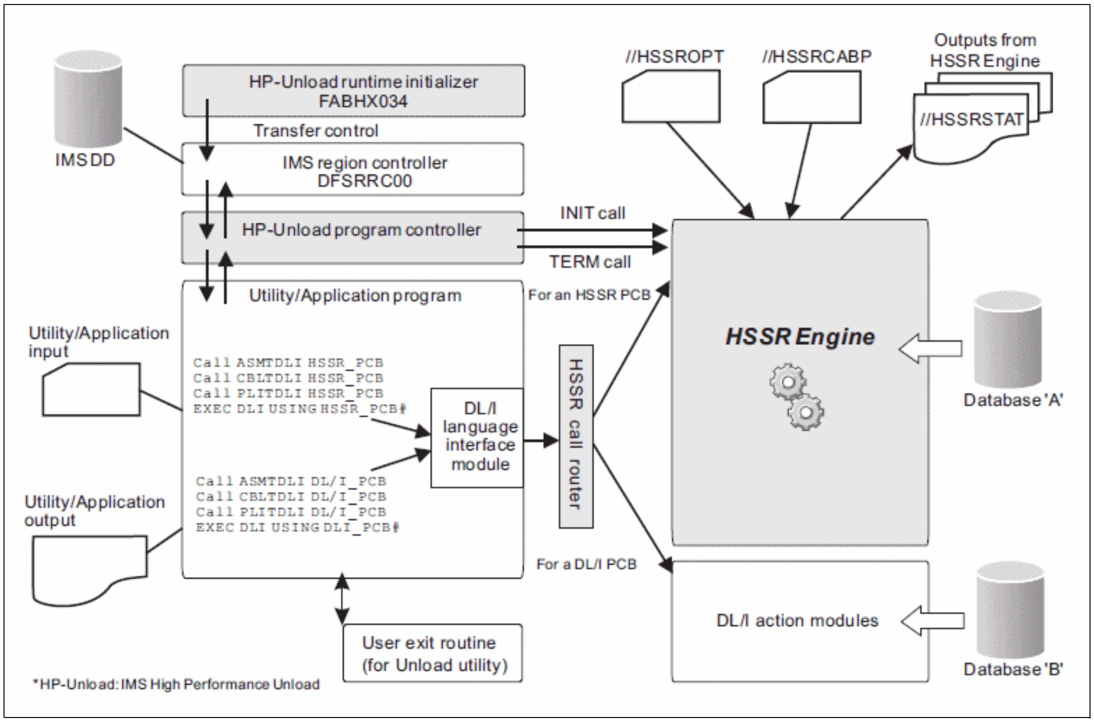


Figure 12-13 Structure of IMS HP Unload and its data flow

For more information, see *IBM IMS High Performance Unload for z/OS Version 1 Release 2 User's Guide*, SC27-0936.

12.5.6 IMS High Performance Load V2.1

IBM IMS High Performance Load for z/OS (which is also referred to as IMS High Performance Load or IMS HP Load) provides a complete set of high performance reorganization reload procedures for the following database organizations:

- HDAM
- HIDAM
- HISAM
- SHISAM
- PHDAM
- PHIDAM

IMS HP Load components

IMS High Performance Load consists of the following components:

- IMS HP Load utility

  Provides high performance reload processing for databases and is a performance replacement of the IMS HD Reorganization Reload utility (DFSURGL0).

- Load API

  By using the Load API, you can run the initial load process much faster than that of IMS DL/I.
- Physical Sequence Sort for Reload utility

  Sorts an unloaded database data set for an HDAM, PHDAM, HIDAM, PHIDAM, HISAM, or SHISAM database to reduce the elapsed time that is required to reload the database.
- Bitmap Resetter utility

  Can be used to adjust the bitmap of an HDAM, HIDAM, PHDAM, or PHIDAM database to accommodate denser packing of the database blocks.

Figure 12-14 shows the general data flow for the IMS HP Load utility.

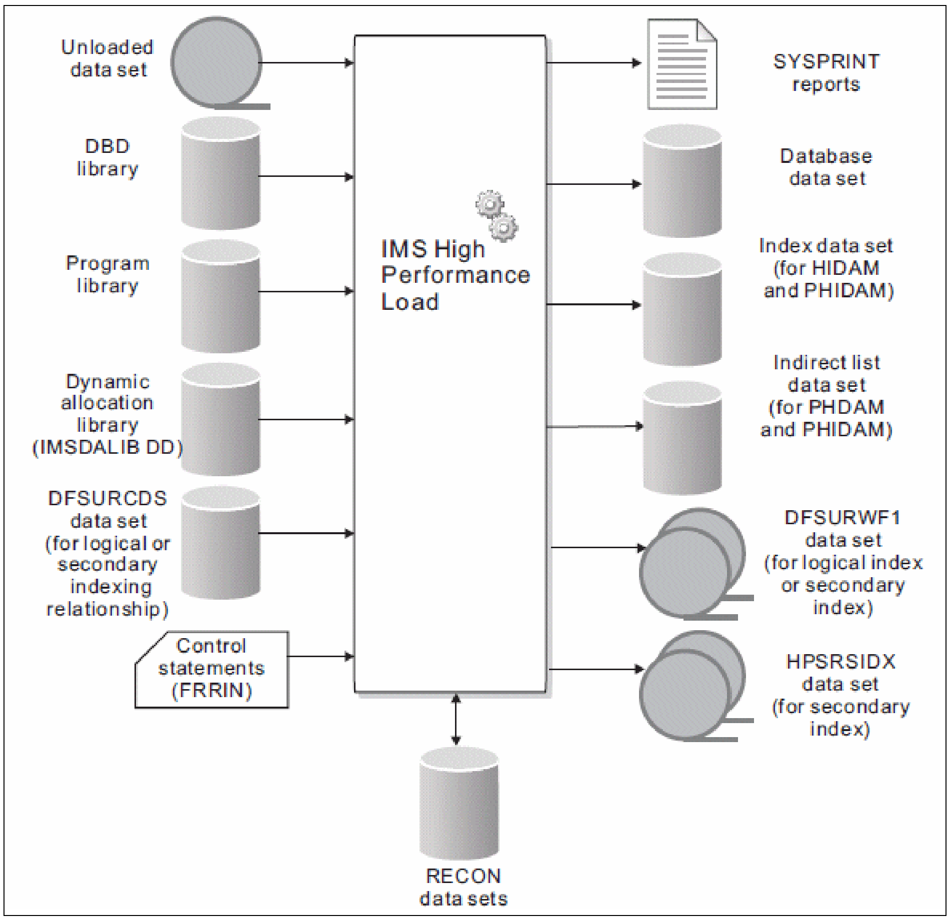


Figure 12-14 General data flow for the IMS HP Load utility

For more information, see *IBM IMS High Performance Load for z/OS Version 2 Release 1 User's Guide*, SC18-9222.

12.5.7 IMS High Performance Prefix Resolution V3.1

IBM IMS High Performance Prefix Resolution for z/OS (which is also referred to as IMS HP Prefix Resolution) is an IMS tool with which you can resolve and update prefixes of IMS databases that are involved in logical relationships as a single job step.

IMS HP Prefix Resolution consists of the following services and utilities:

- A data transfer service that is called HPPRPIPE is used to eliminate the intermediate Work File 2 (WF2) and Work File 3 (WF3) data sets.
- The Prefix Resolution function or the Prefix Update function can be used as a replacement for the IMS Database Prefix Resolution utility or the IMS Database Prefix Update utility.

IMS HP Prefix Resolution functions

IMS HP Prefix Resolution features the following two functions:

- Prefix Resolution

  This function can be used in place of the IMS Database Prefix Resolution utility (DFSURG10) and processes the DFSURWF1 data set, which is generated by the work data set generator (DFSDSEH0) and Index Maintenance (DFSDXMT0) IMS modules.

  This function also creates a data set that contains the information that is needed to resolve the logical relationship pointers that are defined for the databases. It is also used to create a data set that contains information that is needed to create secondary index databases.

- Prefix Update

  This function can be used in place of the IMS Database Prefix Update utility (DFSURGP0). It can be run as a stand-alone job or concurrently during the Prefix Resolution function (which is called the Concurrent Prefix Update). With the use of HPPR buffer handler, it improves the performance considerably over the IMS Database Prefix Update utility.

  If DBRC is active, the Prefix Update function issues a `NOTIFY.REORG` command for each database that was updated.

Figure 12-15 shows the general data flow for Concurrent Prefix Update.

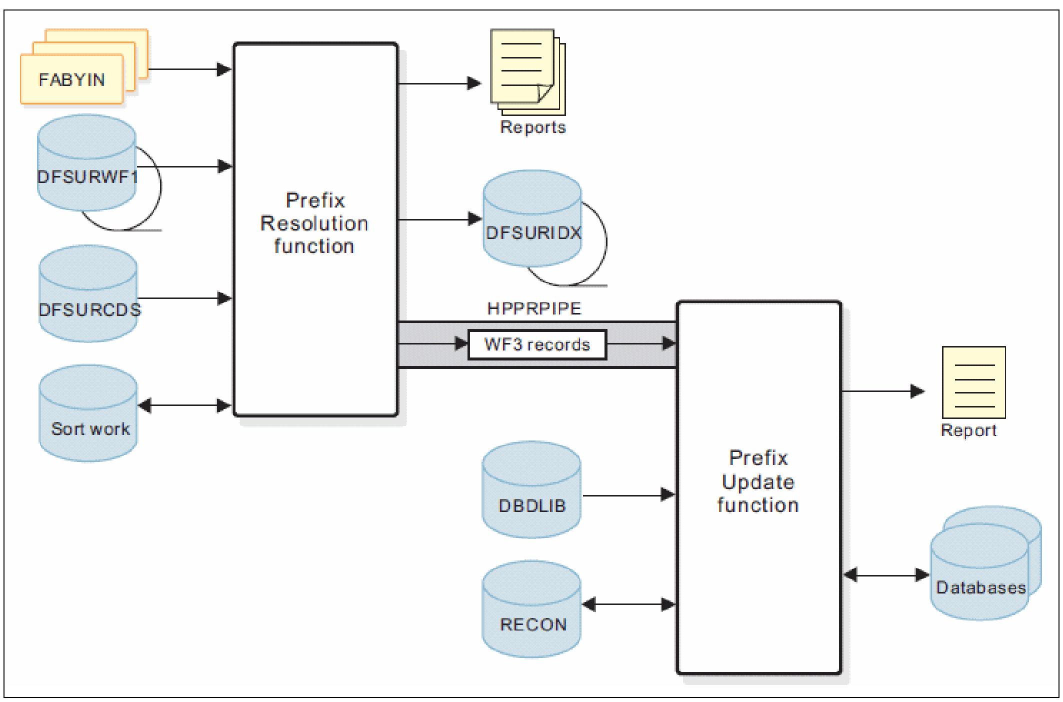


Figure 12-15 General data flow for Concurrent Prefix Update

For more information, see *IBM IMS High Performance Prefix Resolution for z/OS Version 3 Release 1 User's Guide*, SC18-9230.

12.5.8 IMS Index Builder V3.1

By using IBM IMS Index Builder for z/OS (which is also referred to as IMS Index Builder), you can build (or rebuild) IMS secondary indexes, Hierarchical Indexed Direct Access Method (HIDAM) primary indexes, and Indirect List Data Sets (ILDS).

It supports full-function non-partitioned databases and partitioned high availability large databases (HALDB).

Note: References to HIDAM databases also apply to Partitioned Hierarchical Indexed Direct Access Method (PHIDAM) databases. Likewise, references to HDAM databases also apply to Partitioned Hierarchical Direct Access Method (PHDAM) databases.

IMS Index Builder benefits

The tool creates multiple indexes in one job step with which you can rebuild IMS indexes rather than recover or reorganize them by using the traditional unload and reload processes that are used for the primary data store.

The tool eliminates the need to image copy indexes, which means faster recovery and reorganization times. Also, new indexes can be added quickly without reorganizing the primary databases.

For any supported hierarchical direct (HD) database and for HISAM databases, IMS Index Builder scans the existing physical database and builds IMS primary and secondary indexes.

For non-partitioned databases, it creates one or more new secondary index databases. A full database reorganization and initial HALDB load are no longer necessary.

IMS Index Builder components

IMS Index Builder is composed of the driver and the common services group.

The driver is started by the job-step executable file that starts the IMS Index Builder master address space (MAS), analyzes the input, processes the database control tables, and attaches common services.

When the driver receives the messages that indicate the processes are complete, it ends.

The IMS Index Builder code is object-oriented. It is based on the message flow between servers that are attached by the driver after a preliminary analysis of the input control statements.

Figure 12-16 on page 397 shows the IMS Index Builder architecture.

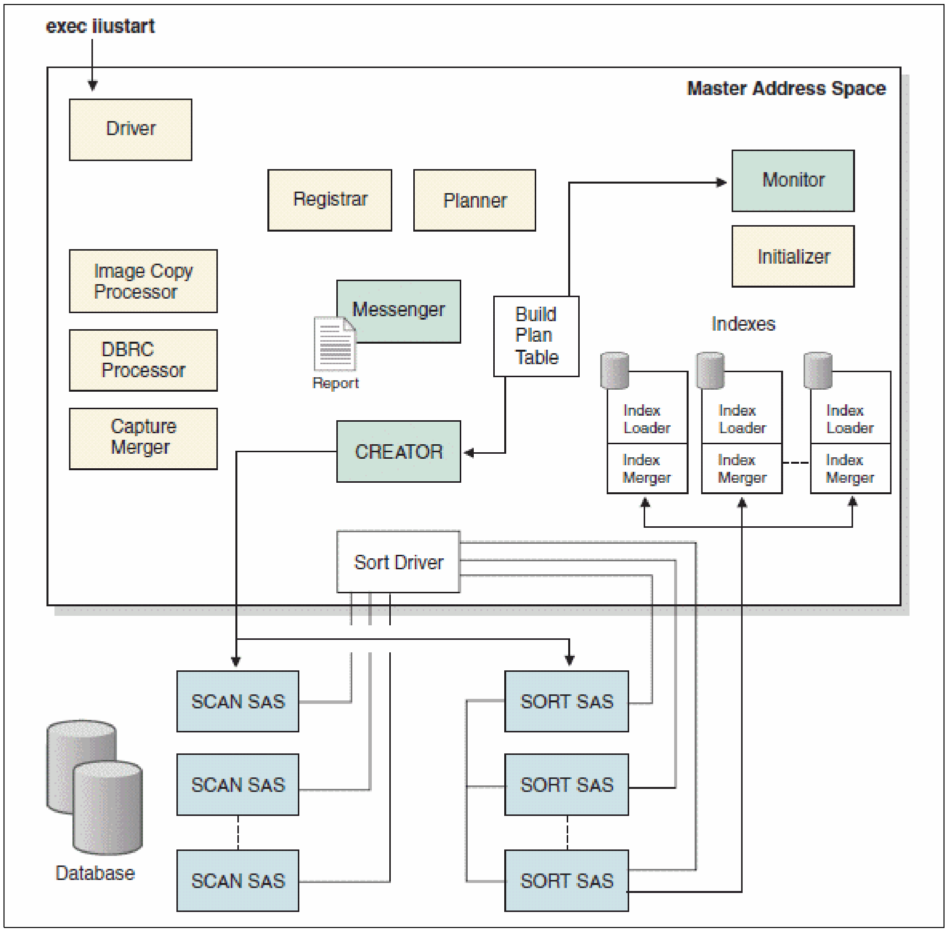


Figure 12-16 IMS Index Builder architecture

For more information, see *IBM IMS Index Builder for z/OS Version 3 Release 1 User's Guide*, SC18-9101.

12.5.9 IMS High Performance Image Copy V4.2

IMS High Performance Image Copy creates image copies of a database and recovers a physically damaged data set of an IMS database. It provides fast back up and recovery of database data sets by using advanced copy technology.

For more information, see 12.4.7, "IMS High Performance Image Copy for z/OS V4.2" on page 383.

12.5.10 IMS High Performance Pointer Checker V3.1

IMS HP Pointer Checker provides utilities that are designed to help ensure that IMS databases are operational, tuned, repaired, and ready for use.

IMS HP Pointer Checker utilities

IMS HP Pointer Checker provides the following utilities:

- HD Pointer Checker

  This utility detects and reports problems of direct or other types of pointers.

- HD Tuning Aid

  This utility produces reports that describe the distribution of root segments in HDAM, HIDAM, PHDAM, or PHIDAM databases.
- DB Historical Data Analyzer

  This utility helps you to analyze the status and historical trend of IMS full-function database data sets that HD Pointer Checker supports.
- Space Monitor

  This utility helps you to forecast potential space usage problems of IMS full-function database data sets that HD Pointer Checker supports and OS data sets (including VSAM data sets).
- DB Segment Restructure

  This utility changes the format of a segment data within any existing full-function database, including HALDB. Its main function is to modify databases in ways that exceed the capabilities of the standard IMS utilities.

Figure 12-17 shows the data flow for HD Pointer Checker.

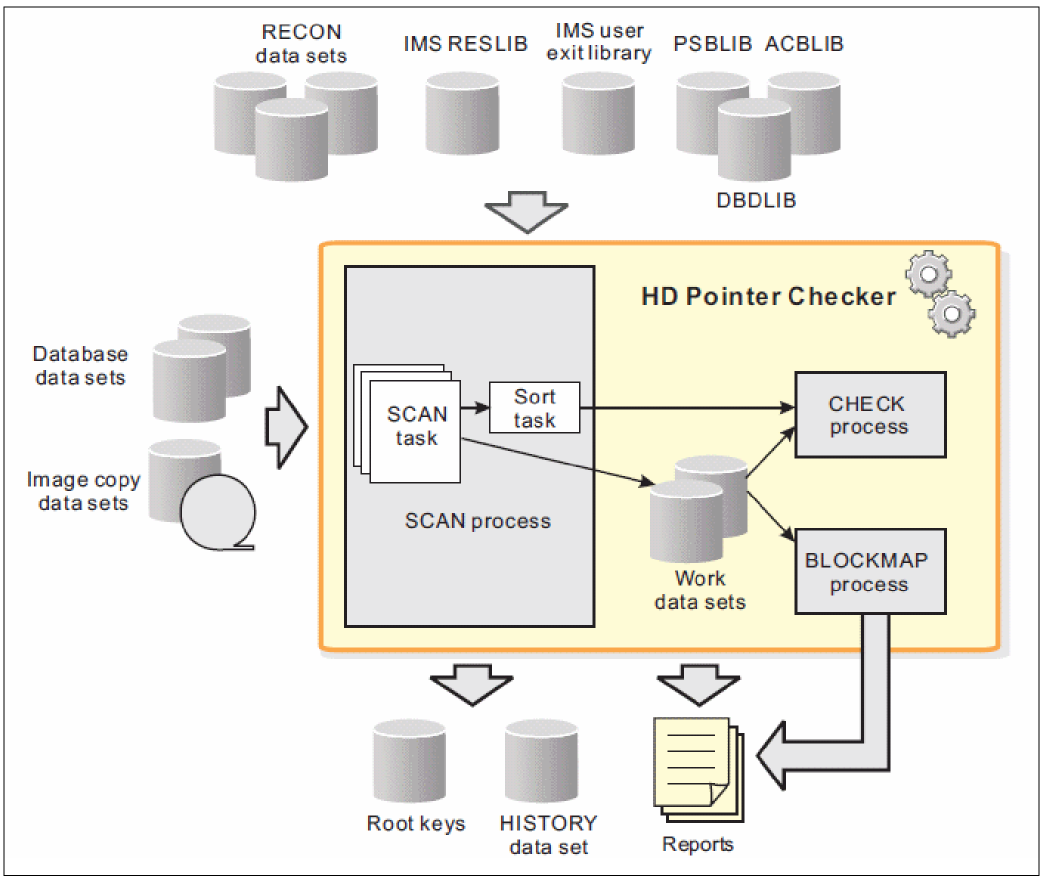


Figure 12-17 Data flow for HD Pointer Checker

IMS HP Pointer Checker enhancements

During changing partition, the reorganization number can be regressed erroneously when Reorg Number Verification is not activated.

If the reorg number is regressed, HALDB ILK (Indirect List Key) can be duplicated, which causes data loss or data corruption.

DUPILKCHK=YES parameter

HD Pointer Checker checks HALDB ILKs and Partition Reorg Number with **DUPILKCHK=YES** parameter to check the following partition issues:

- Duplicate ILKs
- Potential duplicate ILKs
- Incorrect partition reorg number

REPAIRILK=YES parameter

Duplicate ILKs and potential error ILKs can be corrected by using the following HD Pointer Checker utilities:

- Run the HD Pointer Checker utility with the **REPAIRILK=YES** parameter that is specified to gather information about those ILKs in the FABPILK data set.
- Run the ILK Repair utility with specifying the FABPILK data set as input.

For more information, see *IBM IMS High Performance Pointer Checker for z/OS Version 3 Release 1 User's Guide*, SC19-2401.

12.5.11 IMS Database Repair Facility

IMS Database Repair Facility is a powerful tool that is used to repair VSAM and OSAM organized IMS databases that contain pointer or data errors.

For more information, see 12.4.9, "IMS Database Repair Facility" on page 387.

12.5.12 IMS Library Integrity Utilities V2.2

IMS Library Integrity Utilities (IMS LIU) aids you in managing data for the libraries, such as the DBD libraries, PSB libraries, ACB libraries, and RECON data sets that you use when referring to IMS databases.

For more information, see 12.4.8, "IMS Library Integrity Utilities for z/OS V2.2" on page 385.

12.5.13 IMS High Availability Large Database Toolkit

IBM IMS Database Solution Pack for z/OS IMS High Availability Large Database (HALDB) Toolkit (which is also referred to as IMS HALDB Toolkit) provides application-enabling features and system utilities for the improved management and operation of the IMS HALDB environment.

Performance can be improved by the integration of IMS High Performance Tools.

IMS HALDB Toolkit functions

IMS HALDB Toolkit provides the following functions:

- Simplified conversion from non-partitioned FF databases to HALDB:
  - Single-step batch job
  - ISPF Interface
  - Supports conversion from user partitioning database to HALDB
- HALDB maintenance, modeling, and analysis functions, including:
  - Splitting a partition
  - Consolidating partitions

  - Adding an empty partition to the end of a database
  - Merging HALDBs
  - Maintaining PSINDEX

IMS HALDB Toolkit uses the following included tools as needed:

- IMS High Performance Image Copy for z/OS
- IMS High Performance Load for z/OS
- IMS High Performance Pointer Checker for z/OS
- IMS High Performance Unload for z/OS
- IMS Library Integrity Utilities for z/OS
- IMS Online Reorganization Facility

For more information, see *IBM IMS Database Solution Pack for z/OS Version 2 Release 1 IMS High Availability Large Database Toolkit User's Guide*, SC19-4102.

12.5.14 IMS Online Reorganization Facility

IBM IMS Database Solution Pack for z/OS IMS Online Reorganization Facility (which is also referred to as IMS Online Reorganization Facility or ORF) performs a one-step reorganization of IMS databases with a minimal affect on database availability.

ORF supports internal logical relationships, secondary indexes, and HALDBs and it reorganizes multiple HALDB partitions in a single job step.

IMS ORF functions

IMS Online Reorganization Facility allows a database to be updated while the reorganization or unload process is occurring. After all changes are captured and applied, the database is taken offline for a brief period at the end of the reorganization.

During this period, the shadow data sets that were reorganized are renamed to match the original database data set names. A shadow data set is a temporary copy of an original data set.

After the online reorganization is complete, no manual intervention is required, unless the database state is undetermined or if you specified that the shadow database data sets not be renamed.

If ORF job fails before its takeover process completes, a clean-up process restores the status of database and RECON flags to the original state.

If ORF job fails during the takeover phase, the RECON PROHIBIT AUTH flag is kept on. Also, you can resume the takeover process by using the Takeover Restart function of ORF after the problem is resolved.

IMS Online Reorganization Facility supports the following database organization types:

- HDAM
- HIDAM
- HISAM
- SHISAM
- PHDAM
- PHIDAM
- PSINDEX
- Secondary index

Figure 12-18 shows a database reorganization by IMS Online Reorganization Facility.

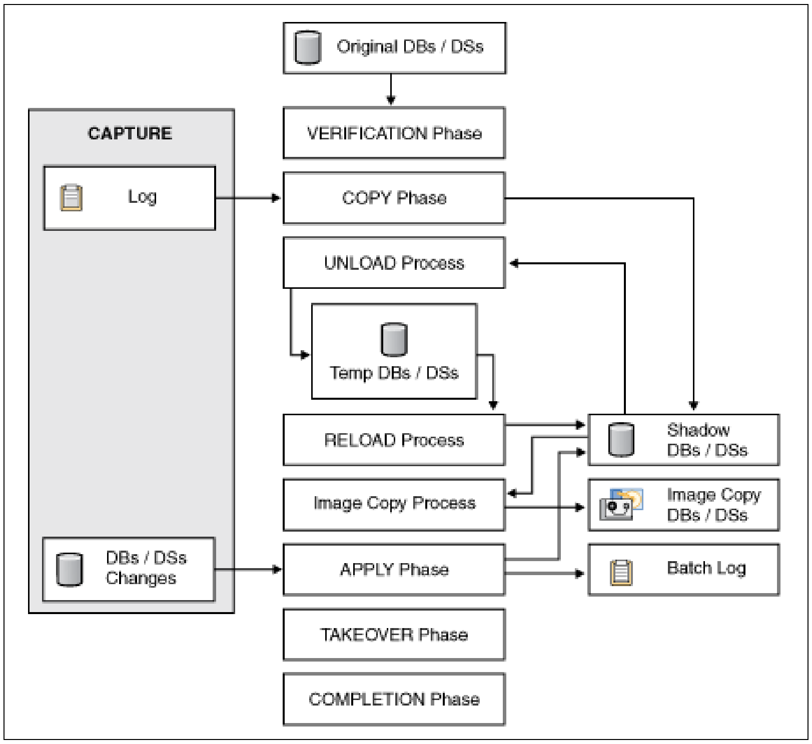


Figure 12-18 Database reorganization by IMS Online Reorganization Facility

For more information, see the following publications:

- *IBM IMS Database Solution Pack for z/OS Version 2 Release 1 IMS Online Reorganization Facility User's Guide*, SC19-4103
- *IBM Tools Base for z/OS Version 1 Release 4 IMS Tools Common Services User's Guide*, SC19-3767

12.6 IMS Recovery Solution Pack for z/OS V1.1

The IBM IMS Recovery Solution Pack for z/OS is software that helps reduce operational complexity and the effect of database backup and recovery on system resources. The solution pack reduces this effect by combining various IBM products into a consolidated solution.

This combination of tools can be used for the simultaneous backup and recovery of multiple data sets and Fast Path areas.

12.6.1 IMS Recovery Solution Pack V1.1 (5655-V86)

IMS Recovery Solution Pack is a product that works with IMS Database Recovery Facility: Extended Functions to simplify your database recovery process by eliminating the need to run a separate recovery job for each database that requires recovery. It includes the functions of the following products:

- IMS Database Recovery Facility V1.1
- IMS High Performance Change Accumulation V1.1

In addition, IMS Database Recovery Facility can interface automatically with the following IMS Tools and utilities during the recovery process:

- IMS Tools Online System Interface
- IMS Index Builder (the IMS DFSPREC0 utility) V3.1 (5655-R01)
- IMS High Performance Image Copy V4.2 (5655-N45) tool
- DEDB Pointer Checker (from IBM IMS High Performance Fast Path Utilities for z/OS (5655-W14), IMS Basic Fast Path Tools) utility

12.6.2 IMS Database Recovery Facility V1.1

IBM IMS Recovery Solution Pack for z/OS: IMS Database Recovery Facility (which is also referred to as IMS Database Recovery Facility) is an IMS tool with which you can simultaneously recover multiple database data sets, HALDB partitions, and Fast Path areas.

It is one of several IBM tools that can help you manage backup and recovery operations for your IMS databases if a system outage or application failure occurs.

IMS Database Recovery Facility functions

By using the IMS Database Recovery Facility, you can simultaneously recover multiple database data sets, HALDB partitions, and Fast Path areas in a single job step.

It also integrates with several other IMS Tools products as integrated auxiliary utilities with which you can create image copies, rebuild indexes, and validate recovered databases.

Running IMS Database Recovery Facility does not require the presence of an active IMS DB/TM or DBCTL subsystem.

You can use the IMS Database Recovery Facility to recover the following database types:

- Full Function: HDAM, HIDAM, HISAM, SHISAM, and INDEX
- HALDB: PHIDAM, PHDAM, and PSINDEX
- Fast Path: DEDB: ADS and MADS

IMS Database Recovery Facility components

The following components are related to the IMS Database Recovery Facility:

- Base Primitive Environment (BPE)

  A set of system services that are needed by the IMS Database Recovery Facility.

- Database Recovery Control (DBRC)

  A facility that maintains records of recovery-related resources in the RECON data sets.

- IMS Database Recovery Facility master address space (MAS).

  An MVS address space in which the controlling components of IMS Database Recovery Facility kept. The log and change accumulation data sets are read in this address space.

- IMS Database Recovery Facility sort subordinate address space or spaces (RSS).

  One or more MVS address spaces that are started by the master address space. These address spaces sort log and change accumulation data and restore image copy data.

Figure 12-19 on page 403 shows the components and the processes of the IMS Database Recovery Facility for the basic product.

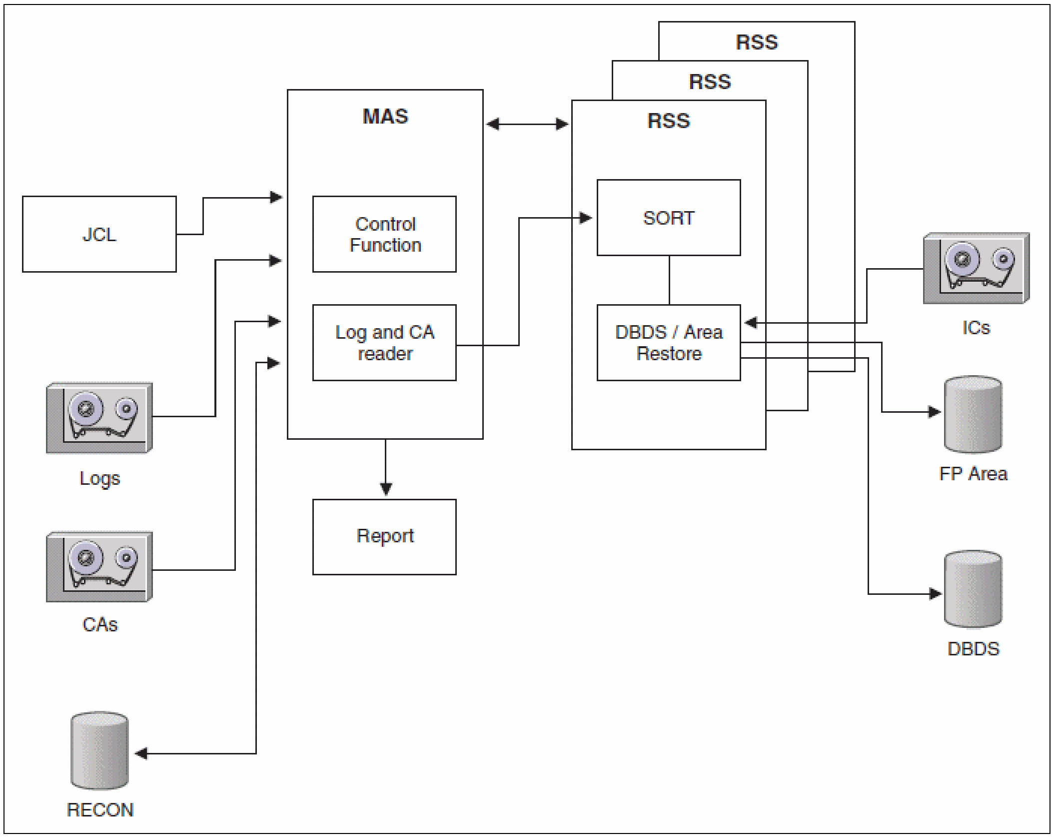


Figure 12-19 Components and processes of IMS Database Recovery Facility for basic product

For more information, see *IBM IMS Recovery Solution Pack for z/OS Version 1 Release 1 IMS Database Recovery Facility User's Guide and Reference*, SC19-2903.

12.6.3 IMS High Performance Change Accumulation V1.1

IMS High Performance Change Accumulation Utility (which is also referred to as IMS HP Change Accumulation Utility) is an IMS tool that improves the performance of change accumulation merge operations by running multiple change accumulation groups in parallel and streaming the output to each group simultaneously.

By using this utility, you can extract database updates with one pass of the data, which minimizes the need for manual intervention.

By using your existing DBRC GENJCL statements, IMS HP Change Accumulation Utility can automatically generate the required JCL.

IMS CA limitations

IMS change accumulation (IMS/CA) has the following limitations:

- IMS/CA is sequential and can process only one CAGRP.
- The input logs (SLDS, RLDS, and OLDS) must be read for each change accumulation operation (repeated I/O).
- When you use DFSUCUM0 to build multiple change accumulation data sets from the same IMS logs, you must process in sequence as many jobs or job steps as there are change accumulation data sets.

IMS HP CA enhancements

IMS HP Change Accumulation Utility augments the existing IMS/CA process by providing an environment on which DFSUCUM0 processes run in parallel. Consider the following points:

- It creates multiple tasks in a single address space for reading the input log data sets and multiple z/OS address spaces for creating the change accumulation data sets.
- In an IMS environment with multiple systems, you can run IMS HP Change Accumulation Utility simultaneously for all systems.
- IMS HP Change Accumulation Utility allows multiple CAGRPs to be processed while reading the input logs in one pass.
- It enables multiple CAGRPs to be processed in parallel by submitting subordinate address spaces with each running change accumulation for a single CAGRP.
- IMS HP Change Accumulation Utility enables multiple CAGRPs to be processed as separate subtasks in a single address space.

Figure 12-20 shows the change accumulation process flow for an IMS HP Change Accumulation Utility environment that includes two parallel log readers and four parallel Sort tasks.

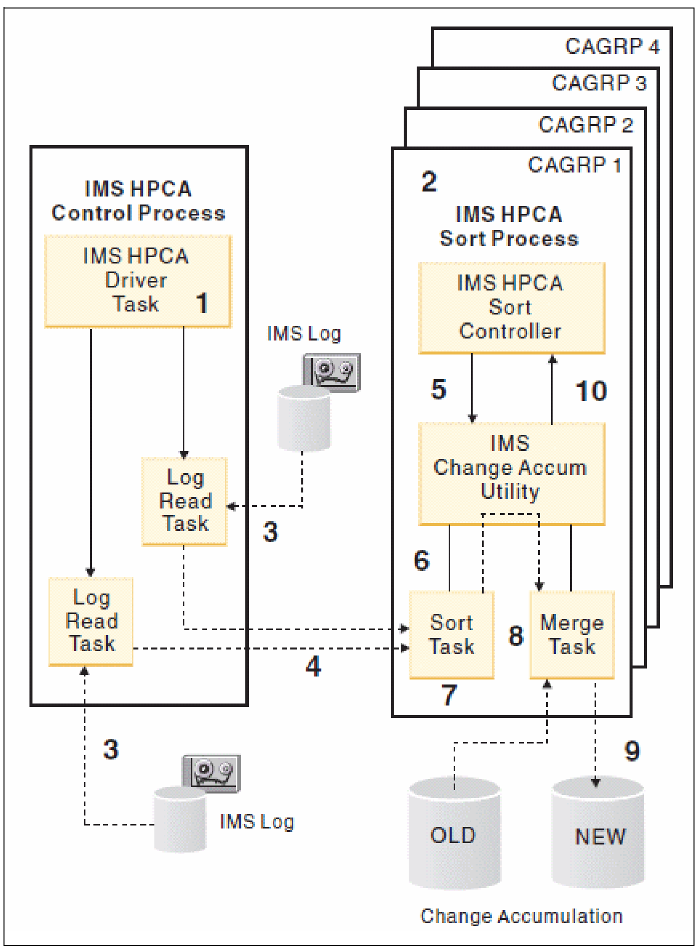


Figure 12-20 IMS HP Change Accumulation Utility architecture and process flow

For more information, see *IBM IMS Recovery Solution Pack for z/OS Version 1 Release 1 IMS High Performance Change Accumulation Utility User's Guide*, SC19-2905.

12.6.4 IMS Index Builder V3.1

By using IMS Index Builder, you can build (or rebuild) IMS secondary indexes, Hierarchical Indexed Direct Access Method (HIDAM) primary indexes, and Indirect List Data Sets (ILDS).

For more information about this product, see 12.5.8, "IMS Index Builder V3.1" on page 396.

12.6.5 IMS High Performance Image Copy V4.2

IMS High Performance Image Copy creates image copies of a database and recovers a physically damaged data set of an IMS database. It provides fast back up and recovery of database data sets by using advanced copy technology.

For more information about this product, see 12.4.7, "IMS High Performance Image Copy for z/OS V4.2" on page 383.

12.7 IMS Performance Solution Pack for z/OS V1.2

The IBM IMS Performance Solution Pack for z/OS tool combines the features and functions of IMS Connect Extensions for z/OS, IMS Performance Analyzer for z/OS, and IMS Problem Investigator for IMS.

The three products are integrated within the IMS Performance Solution Pack, which provides complete analysis of IMS transactions for database analysts.

This combination of tools provides productivity for problem analysts, efficient IMS application performance, improved IMS resource utilization, and high system availability.

12.7.1 Tools in IMS Performance Solution Pack V1.2

The following tools and their product numbers are included in IMS Performance Solution Pack V1.2:

- IMS Connect Extensions V2.3 (5655-S56)
- IMS Performance Analyzer for z/OS V4.3 (5655-R03)
- IMS Problem Investigator for z/OS V2.3 (5655-R02)

12.7.2 Highlights

IMS Performance Solution Pack provides the following features to make end-to-end analysis of IMS transactions faster and easier than ever:

- IMS Applications can have IMS, DB2, WebSphere MQ, IMS Connect in the same transaction
- Tight integration between the three tools
- Supports improved productivity for problem analysts
- Improves IMS application performance
- Enables more efficient IMS resource utilization and higher system availability

IMS Performance Solution Pack provides the following benefits:

- Spend less time and resources troubleshooting IMS performance and system-related problems.

- Avoid performance-related outages and missed service level agreements by making sure that your IMS systems are always running at peak performance.

12.7.3 IMS Connect Extensions V2.3

IBM IMS Connect Extensions for z/OS (referred to as IMS Connect Extensions) is a keytool that enhances the operation of IMS Connect.

A function of IMS, IMS Connect is the premier pathway for accessing IMS applications and databases via TCP/IP.

Benefits

IMS Connect Extensions provide the following key benefits:

- Provides event collection and instrumentation for IMS Connect
- Streamlines operational management of IMS Connect and its clients
- Helps the development of TCP/IP clients and the transition to an SOA
- Supports IMS Connect-to-IMS Connect TCP/IP communications
- Supports Open Database Manager (ODBM)
- Issues IMS Type-1 commands from the ISPF command shell and Operations Console
- Analyzes problems and optimizes performance by recording details of activity in IMS Connect
- Simplifies the deployment of WebSphere and SOAP clients
- Creates detailed IMS Connect reports with IBM's IMS Performance Analyzer for z/OS
- Solves IMS Connect problems with IBM's IMS Problem Investigator for z/OS
- Views graphical real-time reports of TCP/IP activity with IBM's OMEGAMON® XE for IMS on z/OS

Its principal users are IMS tuning specialists, application developers, and administrators.

IMS Connect Extensions components

IMS Connect Extensions consist of several components. Some components run in the IMS Connect address space and are started by IMS Connect when it starts.

IMS Connect Extensions consist of the following main components:

- ISPF dialog client

  Connects via TCP/IP to one or more IMS Connect Extensions console listeners, which provide centralized monitoring and control of IMS Connect systems across your enterprise.

- Operations Console GUI client

  An Eclipse-based PC application that provides a graphical interface to perform IMS Connect Extensions operations.

- Console listener

  An agent that is running in the IMS Connect address space that listens on a TCP/IP port for connections from IMS Connect Extensions clients.

- Repository

  A VSAM key-sequenced data set (KSDS) that contains configuration data for IMS Connect Extensions.
- Exit manager

  When IMS Connect starts, it starts the IMS Connect Extensions user exit manager.
- Event collector

  Collects IMS Connect events and data about the input and output to every function call to every exit, and includes data that is sent to and returned by custom user exits.
- Status monitor agent

  Provides real-time statistics on message processing activity for a system, its exits, datastores, ODBMs, aliases, and ports, including datastores and ports that are added dynamically in IMS.
- Active session agent

  Provides real-time tracking of active TCP/IP and LOCAL port sessions.
- Publisher interface

  Provides to monitoring applications real-time data about event records and active sessions that are running in an IMS Connect region and configuration information about the IMS Connect system.
- Active journals, archive journals, and the Archive Manager

  Active journals contain IMS Connect event data that is recorded by IMS Connect Extensions.
- Batch command, definition maintenance, and reporting utilities

  IMS Connect Extensions include batch utilities that provide more features, such as log formatting and batch administration.

Figure 12-21 on page 408 shows these IMS Connect Extensions components.

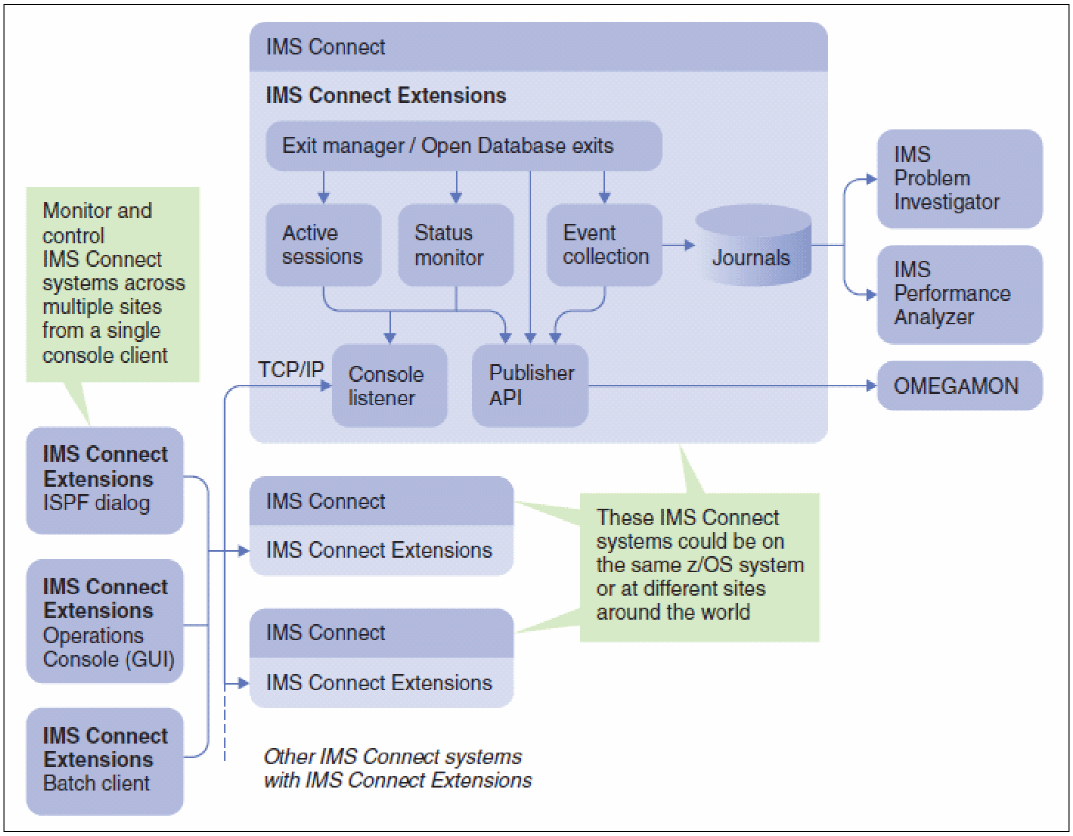


Figure 12-21 IMS Connect Extensions components

For more information, see *IBM IMS Connect Extensions for z/OS Version 2 Release 3 User's Guide*, SC19-3632.

12.7.4 IMS Performance Analyzer for z/OS V4.3

IBM IMS Performance Analyzer for z/OS (which is also referred to as IMS Performance Analyzer) is a performance analysis and tuning aid for DB and TM systems for IMS.

It provides information about IMS system performance for monitoring, tuning, managing service levels, analyzing trends, and capacity planning.

Highlights

IMS Performance Analyzer provides information about IMS system performance for the following functions:

- Tuning and performance
- Problem analysis
- Capacity planning
- Management reporting

IMS Performance Analyzer also provides comprehensive end-to-end reporting, including the following functions:

- Transit analysis for all transaction workloads, including shared-queues by merging sysplex IMS log files

- End-to-end IMS Connect and IMS log reporting, which provides a complete picture of the lifecycle of transactions as they pass-through Connect and into IMS
- IMS monitor, including Fast Path database
- OMEGAMON TRF and ATF

Benefits

IMS Performance Analyzer provides the following benefits:

- Analyze transaction response time.
- Measure usage and availability of important resources, including databases, programs, regions, buffers (including database), and queues (message and other internal queues).
- Plan for IMS operational management, including scheduling database reorganizations, monitoring service-level adherence, charge-back accounting, and capacity planning.
- Monitor significant system events that can adversely affect system performance and availability.
- Boost system and application programmer productivity.
- Report critical performance information, from high-level management summaries to detailed traces for in-depth analysis.
- Analyze the effect of IMS Connect on transaction.

IMS Performance Analyzer operation

IMS Performance Analyzer (IMS PA) provides a comprehensive suite of reports to help you manage the performance and resource utilization of your IMS systems.

It processes IMS Log, Monitor, IMS Connect event data, and OMEGAMON TRF and ATF data to provide comprehensive reports for use by IMS specialists to tune their IMS systems, and by managers to verify service levels and predict trends.

It provides an ISPF-based dialog for you to use to create and maintain your report and extract requests, and generate the JCL to run them by using your specified systems and data files. The IMS PA Control Data Sets are used to store your report and extract requests.

From IMS Log data, IMS PA provides comprehensive information about transit times (actual system performance time), and IMS resource usage and availability.

Extracts of transit time by time interval data can be created from log files, then graphed or exported (with transfer to PC) by using IMS PA facilities.

Report Forms can be used to tailor transit summary and list reports to include only the data fields of interest.

From Monitor data, IMS PA creates summary and analysis reports for regions, resources, programs, transactions, databases, and the total system to analyze your IMS system environment.

IMS PA provides comprehensive reporting from the IMS Connect performance and accounting data that is collected by IMS Connect Extensions for z/OS.

IMS PA provides comprehensive reporting of IMS transaction performance and resource usage statistics that are collected by the Transaction Reporting Facility (TRF) for OMEGAMON XE for IMS.

Figure 12-22 shows an overview of IMS PA operation.

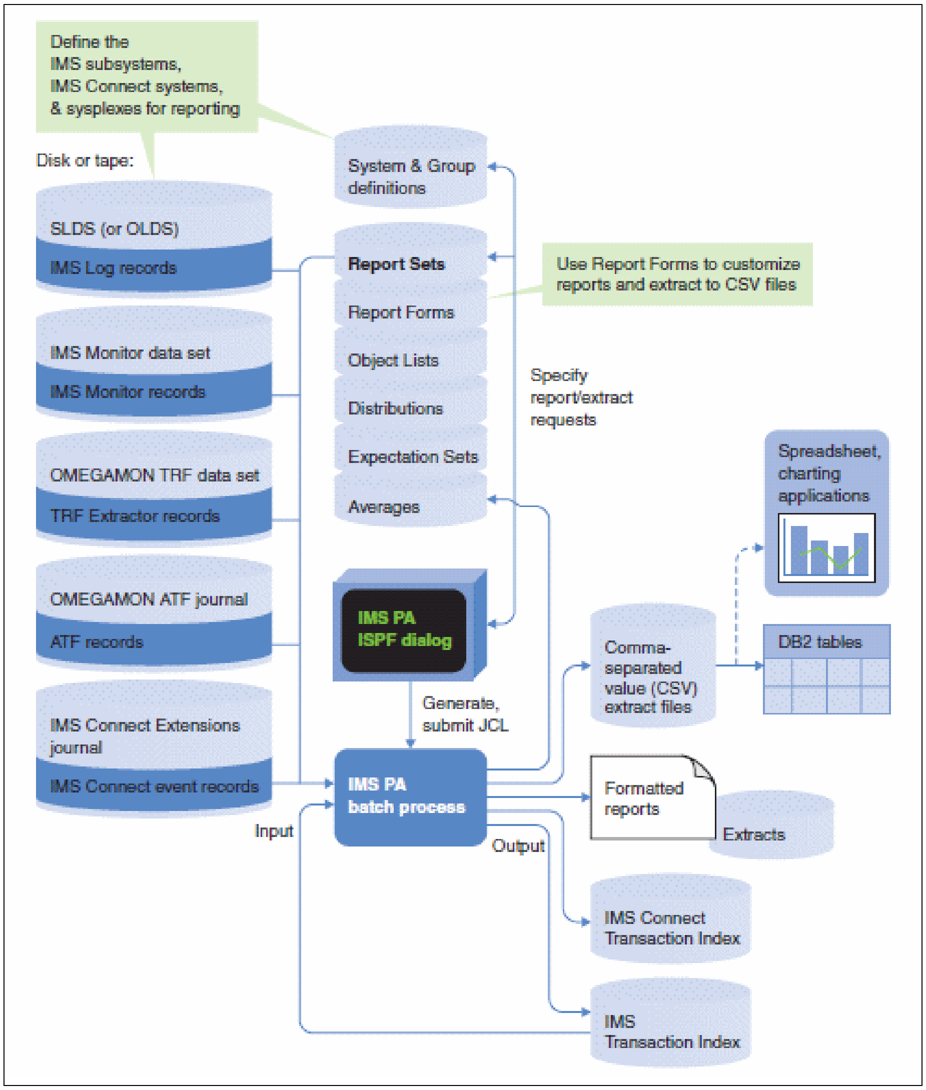


Figure 12-22 Overview of IMS PA operation

For more information, see the following publications:

- *IBM IMS Performance Analyzer for z/OS Version 4 Release 3 User's Guide*, SC19-3633
- *IBM IMS Performance Analyzer for z/OS Version 4 Release 3 Report Reference*, SC19-3634

12.7.5 IMS Problem Investigator for z/OS V2.3

IBM IMS Problem Investigator for z/OS (which is also referred to as IMS Problem Investigator) is a problem analysis tool for Information Management System Database (IMS DB) and Transaction Manager (IMS TM) to help IMS systems and application programmers determine the cause of problems and trace the flow of events end-to-end.

It can help identify the cause of IMS database system problems by providing crucial information, such as who or what incorrectly updated a database, when the database was updated, and how to reverse the changes.

It can also help diagnose IMS transaction management system performance issues by tracking IMS transactions end-to-end through IMS and related systems and determining transaction times and event latencies to help identify bottlenecks.

Highlights

IMS Problem Investigator provides the following core features:

- Formatting: Instantaneously presents and interprets log records.
- Tracking: Track all log records that are associated with a transaction, request, or unit of work with a simple line action.
- Filtering: Use advanced search criteria to find relevant information in the log.
- Merging: Merge multiple logs (even of different types) for a unified view of all activity.
- Reporting and extracting: Output formatted records, CSV, and merged extracts from one or more logs.
- Scrub: Remove sensitive data from log records before making them available for analysis.

Benefits

IMS Problem Investigator provide the following benefits:

- Offers interactive problem determination, with powerful record formatting and navigation aids that help to simplify log analysis.
- Provides an end-to-end replay of an IMS transaction from a single window, including DB2 and WebSphere MQ events.
- Tracks the transaction lifecycle through IMS Connect and into IMS.
- Supports OMEGAMON for IMS Transaction Reporting Facility (TRF) and Application Trace Facility (ATF) for detailed DLI and DB2 call analysis and CPU utilization.
- Provides log record analysis that can drill down to the field-level with online help.
- Displays transaction times and event latencies to help identify bottlenecks.
- Provides a REXX command interface for customized log record analysis and extract.
- Offers batch reporting and extract facility.
- Provides automatic log file selection by using DBRC.

IMS Problem Investigator operation

IMS Problem Investigator enables IMS administrators and programmers to interactively explore formatted, interpreted, and easily customizable views of log records. It also helps users identify and analyze problems quickly, without requiring an expert understanding of log data structures and the relationships between log records.

IMS Problem Investigator supports the following types of log records:

- IMS log
- IMS Transaction Index that is created by IMS Performance Analyzer
- IMS TM and IMS DB monitor data sets
- Common Queue Server (CQS) log stream and extracts
- IMS Connect event data that is collected by IMS Connect Extensions
- IMS Connect Transaction Index that is created by IMS Performance Analyzer
- OMEGAMON Transaction Reporting Facility (TRF) log and extract
- OMEGAMON Application Trace Facility (ATF) journal
- DB2 log
- WebSphere MQ log extract

- SMF - IRLM Long Lock records
- IMS trace table records (67FA, 67FF) in the IMS log
- IMS Repository Audit (FRP) log stream and extracts (introduced in IMS V12)

You can analyze these records through an ISPF dialog, batch reports, and REXX programming services. You also can create filtered extracts and CSV files to aid problem investigation.

Smaller extract files are easier to analyze, but similar efficiencies can be obtained with the original large log files by using time slicing.

You can submit batch requests to format CQS and FRP log streams or create extracts and CSV files. The dialog can format extract files, but not the CQS and FRP log streams directly.

Figure 12-23 shows an interactive ISPF session in which you drill down from the formatted contents of a file to the formatted contents of a record to view the contents of a field with its explained value.

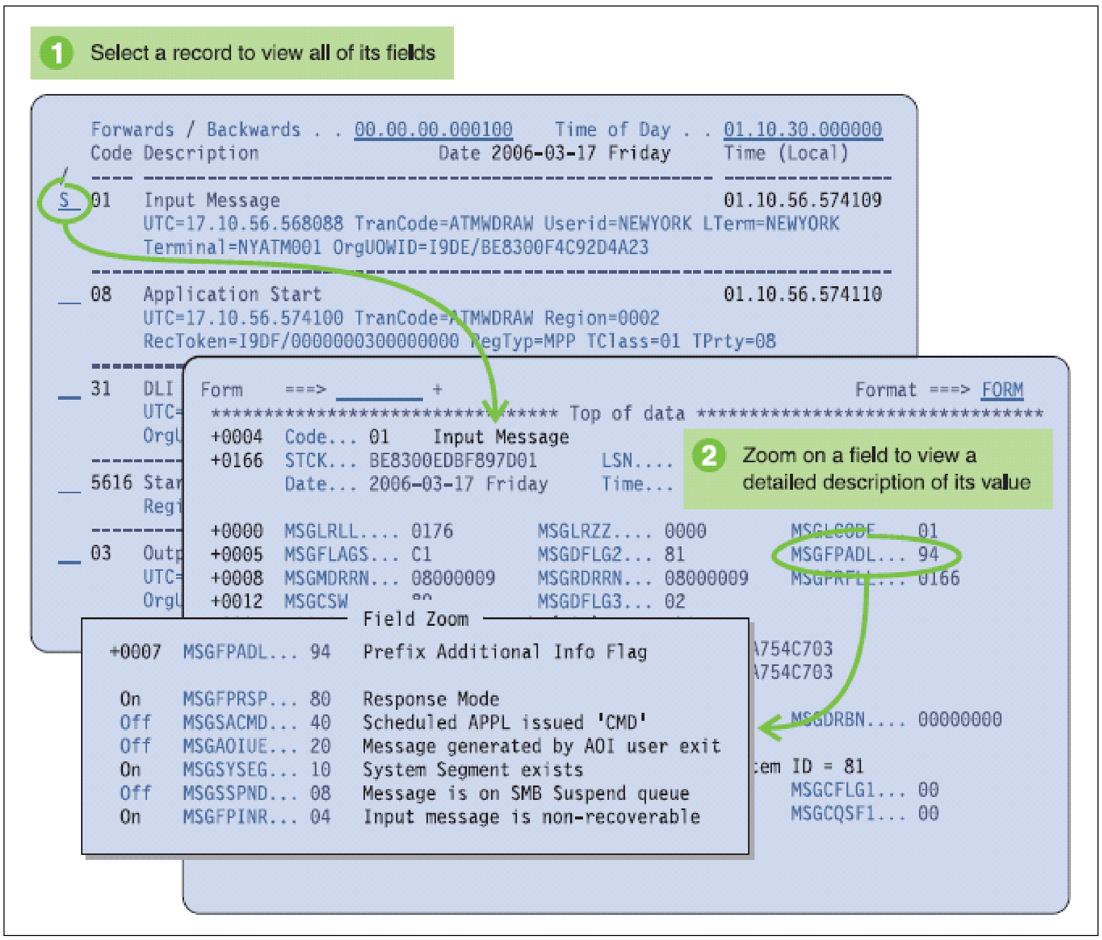


Figure 12-23 View formatted files, records, and fields

For more information, see *IBM IMS Problem Investigator for z/OS Version 2 Release 3 User's Guide,* SC19-3635.

Related publications

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

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some publications that are referenced in this list might be available in softcopy only:

- *IMS Integration and Connectivity Across the Enterprise,* SG24-8174
- *IBM Cognos Business Intelligence 10.2.0 Reporting on IMS,* REDP-5091
- *IMS 12: The IMS Catalog,* REDP-4812-00
- *IMS 12 Selected Performance Topics,* SG24-8071
- *IBM IMS Version 12 Technical Overview,* SG24-7972

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and other materials at this website:

http://www.ibm.com/redbooks

Other publications

The following publications are also relevant as further information sources:

- *IMS Version 13 Application Programming,* SC19-3646
- *IMS Version 13 Communications and Connections,* SC19-3651
- *IMS Version 13 Commands, Volume 1: IMS Commands A - M,* SC19-3648
- *IMS Version 13 Commands, Volume 2: IMS Commands N - V,* SC19-3649
- *IMS Version 13 Commands, Volume 3: IMS Component and z/OS Commands,* SC19-3650
- *IMS Messages and Codes, Volume 1: DFS Messages,* GC18-9712-11
- *IMS Messages and Codes, Volume 2: Non-DFS Messages,* GC18-9713-11
- *IMS Messages and Codes, Volume 3: IMS Abend Codes,* GC18-9714-11
- *IMS Messages and Codes, Volume 4: IMS Component Codes,* GC18-9715-11
- *IMS Version 13 Database Administration,* SC19-3652
- *IMS Version 13 Diagnosis,* GC19-3654
- *IMS Version 13 Database Utilities,* SC19-3653
- *IMS Version 13 Exit Routines,* SC19-3655
- *IMS Version 13 Installation,* GC19-3656
- *IMS Version 13 Operations and Automation,* SC19-3657
- *IMS Version 13 Release Planning,* GC19-3658
- *IMS Version 13 System Definition,* GC19-3660

- *IMS Version 13 System Administration,* SC19-3659
- *IMS Version 13 System Definition,* GC19-3660
- *IMS Version 13 System Utilities,* SC19-3662
- *Program Directory for Information Management System Transaction and Database Servers V13.0*, GI10-8914
- *CICS Transaction Server for z/OS Version 5 Release 1 Resource Definition*, SC34-2868
- *z/OS Communications Server: IP Configuration Reference,* SC27-3651
- *MVS Initialization and Tuning Reference Manual*, SA22-7592
- *z/OS Language Environment Programming Guide,* SA22-7561
- *IBM Tools Base for z/OS Version 1 Release 4 Administration Console User's Guide*, SC19-3764
- *IBM Tools Base for z/OS Version 1 Release 4 Autonomics Director User's Guide*, SC19-3765
- *IBM Tools Base for z/OS Version 1 Release 4 IMS Tools Knowledge Base User's Guide*, SC19-3768
- *IBM Tools Base for z/OS Version 1 Release 4 IMS Tools Common Services User's Guide*, SC19-3767
- *IBM Tools Base for z/OS Version 1 Release 4 Policy Services User's Guide*, SC19-3770
- *IBM Tools Base for z/OS Version 1 Release 4 Distributed Access Infrastructure User's Guide*, SC19-3771
- *IBM Tools Base for z/OS Version 1 Release 4 IMS Hardware Data Compression Extended User's Guide*, SC19-3769
- *IMS Fast Path Solution Pack for z/OS V1.3: Overview and Customization*, SC19-4009
- *IBM IMS Fast Path Solution Pack for z/OS Version 1 Release 3 IMS High Performance Fast Path Utilities User's Guide*, SC19-4010
- *IBM IMS High Performance Image Copy for z/OS Version 4 Release 2 User's Guide*, SC19-2756
- *IBM IMS Library Integrity Utilities for z/OS Version 2 Release 2 User's Guide*, SC19-3979
- *IBM IMS Database Solution Pack for z/OS Version 2 Release 1 Overview and Customization*, SC19-4007
- *IBM IMS Database Repair Facility for IMS Solution Packs User's Guide*, SC19-2916
- *IBM IMS Database Reorganization Expert for z/OS Version 4 Release 1 User's Guide*, SC19-1137
- *IBM IMS High Performance Load for z/OS Version 2 Release 1 User's Guide*, SC18-9222
- *IBM IMS Index Builder for z/OS Version 3 Release 1 User's Guide*, SC18-9101
- *IBM IMS High Performance Unload for z/OS Version 1 Release 2 User's Guide*, SC27-0936
- *IBM IMS High Performance Load for z/OS Version 2 Release 1 User's Guide*, SC18-9222
- *IBM IMS High Performance Prefix Resolution for z/OS Version 3 Release 1 User's Guide*, SC18-9230
- *IBM IMS Index Builder for z/OS Version 3 Release 1 User's Guide*, SC18-9101
- *IBM IMS High Performance Pointer Checker for z/OS Version 3 Release 1 User's Guide*, SC19-2401

- *IBM IMS Database Solution Pack for z/OS Version 2 Release 1 IMS High Availability Large Database Toolkit User's Guide*, SC19-4102
- *IBM IMS Database Solution Pack for z/OS Version 2 Release 1 IMS Online Reorganization Facility User's Guide*, SC19-4103
- *IBM Tools Base for z/OS Version 1 Release 4 IMS Tools Common Services User's Guide*, SC19-3767
- *IBM IMS Recovery Solution Pack for z/OS Version 1 Release 1 IMS Database Recovery Facility User's Guide and Reference*, SC19-2903
- *IBM IMS Recovery Solution Pack for z/OS Version 1 Release 1 IMS High Performance Change Accumulation Utility User's Guide*, SC19-2905
- *IBM IMS Connect Extensions for z/OS Version 2 Release 3 User's Guide*, SC19-3632
- *IBM IMS Performance Analyzer for z/OS Version 4 Release 3 User's Guide*, SC19-3633
- *IBM IMS Performance Analyzer for z/OS Version 4 Release 3 Report Reference*, SC19-3634
- *IBM IMS Problem Investigator for z/OS Version 2 Release 3 User's Guide,* SC19-3635

Online resources

The following websites are also relevant as further information sources:

- IBM IMS Ready. Set. Mobile!:

  http://www.ibm.com/software/data/ims/

- How to Code a Synchronous Program Switch in IMS 13:

  http://www.ibmsystemsmag.com/mainframe/administrator/ims/ims_program_switch/

- IMS 13 Performance Evaluation Summary - Reducing the Total Cost of Ownership with Improved Performance:

  http://ibm.co/1jy804r

- IMS tools and IMS 13 support:

  http://www.ibm.com/support/docview.wss?uid=swg21611198

- DB2 and IMS Tools for System z:

  http://www.ibm.com/software/data/db2imstools

- Information Center:

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.ims13.doc.rpg%2Frpg.htm

- Knowledge Center:

  http://www-01.ibm.com/support/knowledgecenter/SSEPH2_13.1.0/com.ibm.ims13.doc.rpg/ims_coex_over.htm

Help from IBM

IBM Support and downloads:

http://ibm.com/support

IBM Global Services

http://ibm.com/services

Index

G

H

I

J

IBM

Redbooks

IMS Version 13 Technical Overview

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

IMS Version 13 Technical Overview

Explore the features and functions of IMS 13

Understand advantages and applicability of IMS 13

Plan for installation of or migration to IMS 13

IBM Information Management System (IMS) provides leadership in performance, reliability, and security to help you implement the most strategic and critical enterprise applications. IMS also keeps pace with the IT industry. IMS, Enterprise Suite 2.1, and IMS Tools continue to evolve to provide value and meet the needs of enterprise customers.

With IMS 13, integration and open access improvements provide flexibility and support business growth requirements. Manageability enhancements help optimize system staff productivity by improving ease of use and autonomic computing facilities and by providing increased availability. Scalability improvements were made to the well-known performance, efficiency, availability, and resilience of IMS by using 64-bit storage.

IBM IMS Enterprise Suite for z/OS V2.1 components enhance the use of IMS applications and data. In this release, components (orderable or downloaded from the web) deliver innovative new capabilities for your IMS environment. They enhance connectivity, expand application development, extend standards and tools for a service-oriented architecture (SOA), ease installation, and provide simplified interfaces.

This IBM Redbooks publication explores the new features of IMS 13 and Enterprise Suite 2.1 and provides an overview of the IMS tools. In addition, this book highlights the major new functions and facilitates database administrators in their planning for installation and migration.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8224-00 ISBN 0738440035