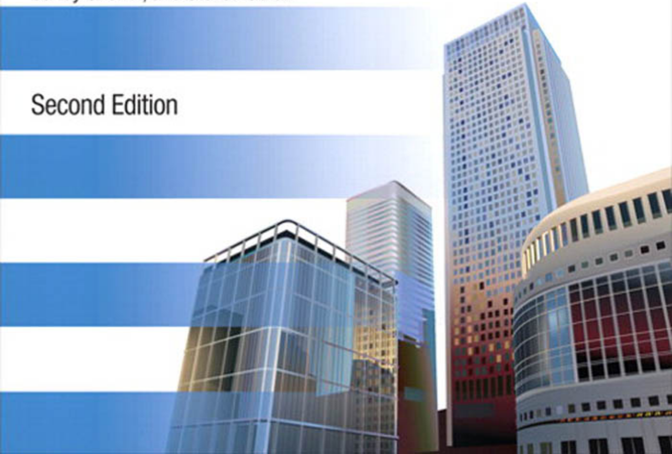
IBM PRESS

An Introduction to IMS

Your Complete Guide to IBM Information Management System

Barbara Klein, Diane Goff, John Butterweck, Kenny Blackman, Margaret Wilson, Moira McFadden Lanyi, Rick Long, Sandy Sherrill, and Steve Nathan

Second Edition



An Introduction to IMS

Second Edition

An Introduction to IMS

Your Complete Guide to IBM Information Management System

Second Edition

Barbara Klein, Diane Goff, John Butterweck, Kenny Blackman, Margaret Wilson, Moira McFadden Lanyi, Rick Long, Sandy Sherrill, Steve Nathan

IBM Press Program Managers: Steve Stansel and Ellice Uffer
IBM Press Consulting Editor: Susan Visser
Cover design: IBM Corporation
Published by Pearson plc
Publishing as IBM Press

Library of Congress Cataloging-in-Publication Data:
An introduction to IMS : your complete guide to IBM's Information Management System / Barbara Klein ... [et al.]. -- [2nd ed.].
p. cm.
ISBN 978-0-13-288687-1 (pbk. : alk. paper)
1. Management information systems. I. Klein, Barbara, 1947-
T58.6.I598 2012
658.4'038011--dc23
2011036854

ISBN-13: 978-0-13-288687-1

ISBN-10: 0-13-288687-1

First printing, December 2011

Contents

Acknowledgments

This book would not be possible without the input and efforts of many people, some of whom are listed here. Our wholehearted thanks go to all who participated in and supported this project, particularly to the following people:

- Dean Meltz, Rick Long, Mark Harrington, Robert Hain, and Geoff Nicholls for the information in the IMS Primer and the first edition of this book, which was the foundation for this second edition
- Susan Visser, Steve Stansel, and Terry Carroll of IBM, and Bernard Goodman, Michelle Housley, and Andy Beaster of Pearson Education, for their guidance and patient assistance in the creation of this book
- Clark Gussin, Visual Designer at IBM, for his assistance with the illustrations in this book
- Don Streicher, Dean Meltz, Suzie Wendler, Kevin Hite, and other members of the IBM IMS organizations for their input to the development of this edition of the book

About the Authors

- Barbara Klein, responsible for the strategy and introduction of new IMS capabilities, has held various positions in IBM Planning, Design, Development, Assurance, Systems Engineering, Marketing, and Management.
- Diane Goff, IBM IMS Advanced Technical Skills Senior I/T Specialist, has presented IMS topics at technical conferences and seminars, worked with IMS customers, and provided System z hardware and software support.
- John Butterweck, IBM Worldwide IMS Technical Support Team member, specializes in assisting customers with IMS installation and maintenance.
- Kenny Blackman, IBM Certified IT Specialist - IMS Advanced Technical Skills, consults on IMS connectivity, IMS application modernization and System z integration architectures, and presents IMS topics at technical conferences and seminars.
- Margaret Wilson, now retired from IBM, spent more than 20 years of her IBM career working with IMS and IMS Tools, testing IMS software, teaching IMS basics, and marketing IMS Tools.
- Moira McFadden Lanyi has been the Technical Editor, Terminologist, and Information Architect for the IMS Information Development team since 2003, and has also worked as a Visual Designer and Project Manager at IBM.
- Rick Long, IMS Development Level 2 Database Support Team member since 2002, began with IMS in the late 1970s working in various programming and database administration roles.

- Sandy Sherrill, IMS Worldwide Market Manager, has spent more than 20 years working on IMS teams.
- Steve Nathan has 38 years of experience as an IMS developer, application analyst, DBA, systems programmer, and performance tuner. He has worked for IBM in IMS Level 2 Support since 2003.

Preface

IBM® Information Management System (IMS™) is one of the world's premier software products. Period. IMS is not mentioned in the news and, until fairly recently, was barely mentioned in computer science classes. But IMS has been and, for the foreseeable future, will continue to be a crucial component of the world's software infrastructure.

Since its origin with the National Aeronautic and Space Administration (NASA) Apollo program, IMS has provided the foundation that enables government agencies and businesses to access, manipulate, and exploit their vast stores of data. As the information age evolves, so does IMS.

The purpose of this book is twofold:

- To introduce IMS and provide basic education about this cornerstone product
- To reintroduce IMS to the computer science field in general

Prerequisite Knowledge

Before reading this book, you should be able to describe the fundamentals of data processing, including the function of operating systems, access methods, and job control language (JCL). Specifically, you should be familiar with the z/OS® operating system or any of its predecessors. The authors of this book assume that most readers are data processing professionals. You can learn more by visiting the z/OS Basic Skills Information Center at http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp.

Learning More about IMS

This book introduces the fundamentals of IMS. To learn more, including about how to use the product capabilities described in this book, read the IMS product publications, which are available in PDF book and HTML format on the web. Publications for all supported versions of IMS are available in the IMS Information Center at http://publib.boulder.ibm.com/infocenter/imzic.

We Welcome Your Feedback

Your feedback helps us provide the most accurate and highest quality information. If you have comments about this book or any other IMS product information, you can send us feedback by either of the following methods:

- Clicking the Feedback link in the footer of any IMS Information Center topic
- Sending an email to imspubs@us.ibm.com

CHAPTER 1

IMS: From Apollo to Enterprise

IMS is IBM's premier transaction and pre-relational database management system, virtually unsurpassed in database and transaction processing availability and speed. IMS clients have trusted IMS with their most critical business asset—their operational data—for decades.

Today's IMS has only a superficial resemblance to the product that first shipped in 1969. However, an application program that ran on IMS in 1969 will still run today, unchanged, on the current release of IMS. From the beginning, IBM's focus on the future of IMS has been unwavering.

IMS continues to be a strategic component of today's enterprise computing environments. This chapter highlights some of the history of IMS and describes how IMS fits into contemporary IT multitiered enterprise architectures.

In This Chapter

- IMS and the Apollo Program
- IMS as a Database Management System
- IMS as a Transaction Manager
- Who Uses IMS?
- IMS and Enterprise Integration

IMS and the Apollo Program

On May 25, 1961, United States President John F. Kennedy challenged American industry to send an American man to the moon and return him safely to earth, thus launching the Apollo program. North American Aviation, in partnership with IBM, fulfilled the requirement for an automated system to manage large bills of material for the construction of the spacecraft in 1965. In 1966, the IBM and North American Aviation teams were joined by three members from Caterpillar Tractor. Together, they designed and developed a system that was called Information Control System and Data Language/Interface (ICS/DL/I).

The IBM team completed and shipped the first release of ICS in Los Angeles in 1967, and in April 1968, ICS was installed. The first "READY" message was displayed on an IBM 2740 typewriter terminal at the Rockwell Space Division at NASA in Downey, California, on August 14, 1968. Less than a year later, on July 20, 1969, Apollo 11 landed on the moon's surface. ICS was subsequently relaunched as Information Management System/360 (IMS/360) and made available to the IT world. In short order, IMS helped NASA fulfill President Kennedy's dream and also became the foundation for the database management system (DBMS) business.

Much has changed since 1968; IMS continues to evolve to meet and exceed the data processing requirements demanded by today's enterprise businesses and governments.

IMS as a Database Management System

The IMS DBMS introduced the idea that application code should be separate from data. The point of separation was the Data Language/Interface (DL/I). IMS controls the access and recovery of the data. Application programs can still access and navigate through the data by using the DL/I standard callable interface.

This separation established a new paradigm for application programming. The application code could now focus on the manipulation of data, without the complications and overhead associated with the access and recovery of data. This paradigm nearly eliminated the need for redundant copies of the data. Multiple applications could access and update a single instance of data, thus providing current data for each application. Online access to data also became easier because the application code was separated from the data control.

IMS as a Transaction Manager

IBM developed an online component to ICS/DL/I to support data communication access to the databases. The DL/I callable interface was expanded to the online component of the product to enable data communication transparency to the application programs. A message queue function was developed to maintain the integrity of data communication messages and to provide for scheduling of the application programs.

The online component to ICS/DL/I became the Data Communications (DC) function of IMS, which is known today as the IMS Transaction Manager (IMS TM).

Who Uses IMS?

Over 90% of the top worldwide companies in the following industries use IMS to run their daily operations:

- Manufacturing

- Finance
- Banking
- Retailing
- Aerospace
- Communications
- Government
- Insurance
- High technology
- Healthcare

IMS remains a viable, even unmatched, platform to implement very large online transaction processing (OLTP) systems and, in combination with web application server technology, is the foundation for a new generation of web-based, high-workload applications.
The following quote describes how one analyst[1] views IMS:

> IMS is not "old technology," but it is time-tested technology. Its platform, the mainframe, is far from obsolete and in many ways is out in front regarding virtualization and consolidation capabilities that new technologies are still catching up to. For the vast majority of IMS users, the emphasis should be on "How can I expand my use of IMS to take maximum advantage of its potential?"

Here are some interesting facts about how IMS is used.

- **IMS manages a large percentage of the world's corporate data.**
  - Over 90% of top Fortune 1000 companies use IMS.
  - IMS manages over 15 million gigabytes of production data.
  - More than $3 trillion per day is transferred through IMS by one customer.
- **IMS processes over 50 billion transactions per day.**
  - IMS serves more than 200 million users every day.
  - IMS processes over 180 million transactions per day on a single IMS system via IMS Connect for one customer.
  - IMS manages over 375 million accounts for one customer.

1Quote taken from IDC white paper, *A Platform for Enterprise Data Services: The Proven Power and Flexibility of IMS from IBM.*

IMS and Enterprise Integration

Today's IT architecture requires end-to-end integration across the enterprise and with key partners, suppliers, and customers. Such an architecture enables enterprises to respond with flexibility and speed to any customer demand, any market opportunity, and any external threat. This level of IT complexity must be addressed by an infrastructure that meets the following requirements:

- **Based on open standards.** IMS fully supports Java standards for application development and XML for transparent document interchange.
- **Heterogeneous.** IMS applications can be developed on workstations and run in the host environment. IMS applications and data can be accessed from any platform, including Linux®, almost anywhere on the Internet, through the IMS Connect function, the IMS TM Resource Adapter, and the IMS Universal drivers. IMS Enterprise Suite SOAP Gateway enables IMS applications to provide and request web services independent of platform, environment, application language, or programming model.
- **Integrated.** Integration has always been an IMS priority. IMS ships connectors and tooling with IBM WebSphere® solutions so that customers can connect to IMS applications and data by using the tools and connectors of their choice. IMS delivers Eclipse-based plug-ins for use with Rational® Developer for System z® that simplify common application development tasks.
- **Scalable.** IMS continues to address scalability needs by providing the highest possible availability, performance, and capacity.
- **Enabled with self-managing capabilities.** IMS addresses the need to assist technical support staff in being more productive, keeping systems continuously available in an operating environment that is growing increasingly complex.
- **Reliable.** One large IMS customer has operated for more than 10 years (and counting) without an outage in a 24x7 environment. Most IMS customers will say that they cannot recollect when their last unplanned outage occurred.

The ongoing focus that IMS has in support of an open, integrated, simplified, on-demand operating environment, and the success of those who capitalize on their IMS investment, suggests that IMS will remain a major factor in enterprise architectures worldwide and continue to be a great fit for the future of IT.

CHAPTER 2

Overview of the IMS Product

IMS consists of three components:

- IMS Database Manager (IMS DB)
- IMS Transaction Manager (IMS TM)
- A set of system services that provide common services to IMS DB and IMS TM

Often referred to collectively as IMS DB/DC (DC stems from the original name for the IMS Transaction Manager: Data Communications), these components comprise a complete online transaction and database processing environment that provides continuous availability and data integrity. IMS delivers accurate, consistent, timely, and critical information to application programs, which in turn deliver the information to many end users and programs.

IMS has been developed to provide an environment for applications that require very high levels of performance, throughput, and availability. IMS runs on z/OS and on zSeries® hardware, and uses many of the facilities offered by the operating system and hardware.

IMS TM and IMS DB can be ordered separately if both components are not required. The appropriate system services are provided for the components that are ordered. When IMS DB is ordered by itself, it is called DB Control (DBCTL). When IMS TM is ordered by itself, it is called DC Control (DCCTL).

Because each new release of IMS is upwardly compatible, applications that were written for an earlier release of IMS run without modification (in most cases) with a new release. This is one way IMS protects the investment that customers have made in their applications.

To accommodate the changing requirements of IT systems, many new features have been added to IMS over the years. IMS is also always among the first to support new computing paradigms such as service-oriented architecture (SOA) and Web 2.0.

Application programs that use IMS functions can be written in a number of programming languages. These applications access IMS resources by calling a number of IMS functions through standard application programming interfaces (APIs):

- Data Language/Interface (DL/I)
- Java™ Database Connectivity (JDBC)
- Java Message Service (JMS)

In This Chapter

- IMS Database Manager
- IMS Transaction Manager
- IMS System Services
- IMS Product Documentation
- Hardware and Software Requirements for IMS

IMS Database Manager

IMS DB is a database management system (DBMS) that helps you organize business data with both program and device independence.

Hierarchical databases and data manipulation language (DL/I calls) are the heart of IMS DB. Data within the database is arranged in a tree structure, with data at each level of the hierarchy related to, and in some way dependent on, data at a higher level of the hierarchy. Figure 2-1 shows the hierarchical database model. In a hierarchical database, data is stored in the database only once, but it might be able to be retrieved by several paths. IMS logical relationships and secondary indexes are discussed in Part II, "IMS Database Manager."

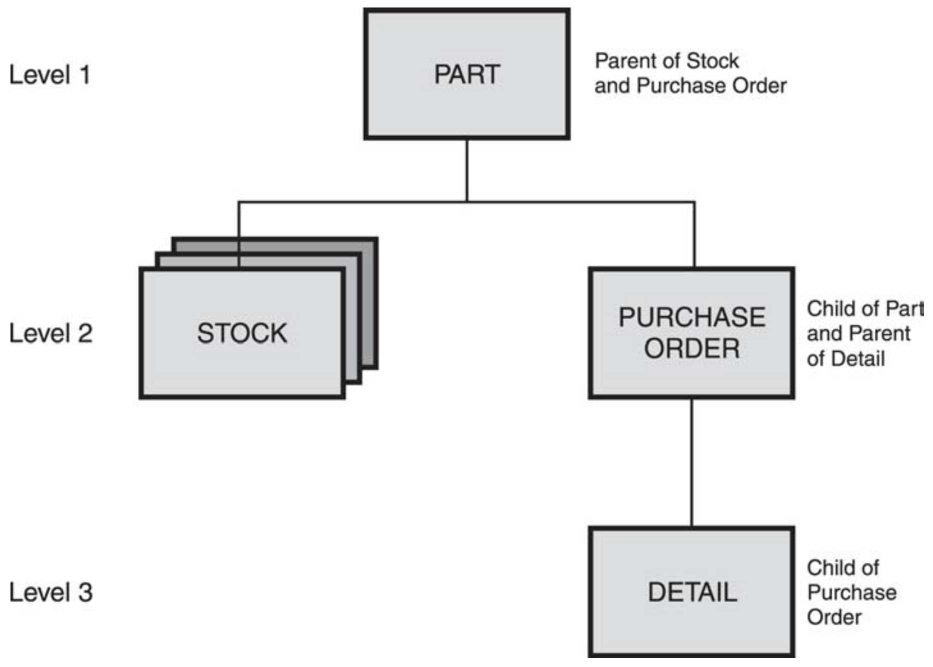


Figure 2-1 Hierarchical Database Model

With IMS DB, you can do the following:

- Maintain data integrity. The data in each database is consistent and remains in the database even when IMS DB is not running.
- Define the database structure and the relationships between the database segments. IMS segments in one database can have relationships with segments in a different IMS database (logical relationships). IMS databases also can be accessed by one or more secondary indexes.
- Provide a central point of control and access for the IMS data that is processed by IMS applications.
- Perform queries against the data in the database.
- Perform database transactions (inserts, updates, and deletes) as a single unit of work so that the entire transaction either completes or does not complete.
- Perform multiple database transactions concurrently, with the results of each transaction kept isolated from the others.
- Maintain the databases. IMS DB provides facilities for backing up and recovering IMS databases, as well as facilities for tuning the databases by reorganizing and restructuring them.

In addition, IMS DB enables you to adapt IMS databases to the requirements of varied applications. Application programs can access common (and therefore consistent) data, thereby reducing the need to maintain the same data in multiple ways in separate files for different applications.

IMS databases are accessed internally using a number of IMS database organization access methods. The database data is stored on disk storage using z/OS access methods. IMS DB provides access to these databases from applications running under the following:

- IMS Transaction Manager (IMS TM)
- CICS® Transaction Server for z/OS
- z/OS batch jobs
- WebSphere Application Server for z/OS
- DB2® for z/OS stored procedures

In addition, IMS Version 11 and later includes the IMS Open Database Manager (ODBM). ODBM enables access to IMS databases from Java programs anywhere in the enterprise using distributed relational database architecture (DRDA) and distributed database management (DDM).

IMS Transaction Manager

IMS TM is a message-based transaction processor.

DEFINITIONS

- A *transaction* is a specific set of input data that triggers the execution of a specific business application program (a process or job). The message that triggers the application program, and the return of any results, is considered one transaction.
- The word *terminal* is used throughout this book to describe devices and controllers. The operator terminals can be keyboard printers, display stations with keyboards, communication terminals, or a combination of these devices.

IMS TM provides services to do the following:

- Process input messages received from a variety of sources, such as the terminal network, other IMS systems, WebSphere MQ, and the web
- Process output messages that are created by application programs
- Provide an underlying queueing mechanism for handling these messages
- Provide interfaces to the TCP/IP network (IMS Connect)
- Provide high-volume, high-performance, high-capacity, low-cost transaction processing for both IMS DB hierarchical databases and DB2 relational databases

IMS TM supports many terminal sessions, on a wide variety of terminals and devices, at extremely high transaction volumes. The users of the terminal sessions can be any of the following:

- People at terminals or workstations
- Other application programs on the same z/OS system, on other z/OS systems, or on non-z/OS platforms

When IMS TM is used with a database manager, IMS TM extends the facilities of the database manager to the online, real-time environment. IMS TM enables terminals, other devices, and other subsystems to enter transactions that initiate application programs that can access IMS DB and DB2 databases and return results.

You can define a variety of online processing options. For example, you can define transactions for high-volume data-entry applications, others for interactive applications, and still others to support predefined queries.

For more information about IMS TM, see Chapter 12, "Overview of the IMS Transaction Manager," and Chapter 13, "How IMS TM Processes Input," in Part III, "IMS Transaction Manager."

IMS System Services

IMS System Services provide the following services to both IMS DB and IMS TM:

- Data integrity
- Restart and recovery of IMS after normal shutdown and failures
- Security, by controlling access to and modification of IMS resources
- Application program management by dispatching work, loading application programs, and providing locking services
- Diagnostic and performance information
- Facilities for operating IMS
- Interfaces to other z/OS subsystems that communicate with IMS applications

The IMS System Services are accessed by doing the following:

- Issuing IMS commands
- Running IMS-supplied utility programs
- Running IMS-supplied or user-written exit routines
- Defining the services that you want as part of the system definition process

IMS Product Documentation

The primary resource for IMS product documentation is the IMS Information Center on the web. IMS publications are available in this Eclipse-based information center in both PDF book and HTML format at http://publib.boulder.ibm.com/infocenter/imzic. A locally installable version of the Information Center can be installed on a computer or intranet server and is available from the IBM Publications Center. See the topic "Installable Information Center" in the IMS Information Center for details.

For customers who are unfamiliar with using the Information Center, animated tutorials are available that describe how to use the features of the Information Center. See the topic “Learn to Use the Information Center” in the IMS Information Center.

Other IMS documentation, such as IBM Redbooks® publications, white papers, presentations from technical conferences, demos, and more, is available on the IMS home page at www.ibm.com/ims.

Hardware and Software Requirements for IMS

This section briefly describes the hardware and software that IMS requires. For complete details about the hardware and software requirements and compatibility of IMS releases with versions of the operating system and associated products, see the *Release Planning* publication for each release of IMS. *Release Planning* publications are available in the IMS Information Center.

Hardware

IMS runs on all IBM processors that are capable of running z/OS. IMS can run on either 64-bit processors or 32-bit processors.

For all system libraries and working storage space, any device that is supported by the operating system is allowed.

For IMS database storage, any device that is supported by the operating system is allowed within the capabilities and restrictions of basic sequential access method (BSAM), queued sequential access method (QSAM), and virtual storage access method (VSAM).

Software

Each release of IMS requires a minimum z/OS level. There are also minimum release levels for the following base elements of z/OS, which cannot be ordered separately:

- Data Facility Storage Management System (DFSMS)
- SMP/E
- JES2
- JES3
- TSO/E

There are also minimum release levels for the following elements:

- IBM High-Level Assembler Toolkit, a separately orderable feature of z/OS.
- z/OS Security Server RACF®, or an equivalent product, if security is used. RACF is available with the IBM SecureWay Security Server for z/OS, a separately orderable feature of z/OS.
- ISPF.

- z/OS Communications Server, if IMS TM is used.
- IRLM (internal resource lock manager).

IMS also operates in a virtual machine (VM) under the control of z/OS. This environment is intended for use in a program development, testing, or non-XRF production environment.

CICS Subsystems Supported

IMS DB can be connected (using the IMS Database resource adapter) to CICS, and IMS TM can be connected (using the appropriate TM interface) to CICS. Minimum CICS levels are documented in the *Release Planning* publications.

DB2 Subsystems Supported

IMS TM can be connected to DB2 products. Supported DB2 products and release levels are documented in the *Release Planning* publications.

IMS Development Language

IMS is written in High Level Assembler, PL/X (an internal IBM development language), C, C++, and Java.

Application Programming Languages Supported

You can write IMS applications in the current versions of the following languages:

- Ada
- COBOL for OS/390 & VM
- Enterprise COBOL for z/OS
- Enterprise PL/I for z/OS
- IBM High Level Assembler for z/OS & Java & z/VSE®
- Java, using the IBM 31-bit SDK for z/OS, Java Technology Edition
- PL/I for z/OS and OS/390
- TSO/E REXX
- VS Pascal
- WebSphere Studio Site Developer
- z/OS C/C++

These products require the IBM Language Environment for z/OS: COBOL for OS/390 & VM, and PL/I for z/OS and OS/390.

CHAPTER 3

Access to and from IMS

Secure access to IMS is achieved by IMS-managed application programs that require the resources provided by the IMS Database Manager (IMS DB), the IMS Transaction Manager (IMS TM), and IMS System Services. These application programs run in IMS-dependent regions and use IMS-supported programming languages to access IMS resources through the DL/I and Java Database Connectivity (JDBC) application programming interfaces (APIs). Secure accesses to IMS-managed resources from application programs that run in other runtime environments use the principles of service-oriented architecture and the components of the IMS SOA Integration Suite. When an end user accesses IMS, both application program types are available.

In This Chapter

- IMS-Managed Application Program Access
- Accessing IMS from Other Application Runtime Environments
- Access to and from IMS Using the IMS SOA Integration Suite Components
- Accessing from IMS
- Accessing to and from IMS

IMS-Managed Application Program Access

For access to IMS resources, application programs managed by IMS invoke Data Language/Interface (DL/I) calls by using the DL/I interface, and invoke SQL calls by using the Java Database Connectivity (JDBC) interface.

Accessing IMS by Using DL/I Calls

DL/I calls are the standard interface for IMS. These calls provide navigational access to business data managed by IMS DB and transparent data communication processing managed by IMS TM. IMS includes a sustainable runtime environment for application programs that have been developed over the years to support the business services for which they were originally designed. As business needs have evolved, IMS application programs have become the base for new business solutions. To illustrate how the IMS architecture supports the evolution of application programs, Figure 3-1 shows an IMS application program running in an IMS dependent region accessing an IMS database through DL/I.

IMS DB maps the DL/I call to the underlying data access methods provided by IMS and z/OS. This mapping enables the application code to function independently of the data access methods and to remain transparent to the physical storage of the data.

When the application program was originally written, it was designed to interface with the IBM Systems Network Architecture (SNA) using DL/I calls for data communication. As new network device types and network protocols emerge, the application program continues to process an IMS transaction message request with minimal or no application code changes required.

Accessing IMS TM

IMS TM supports an SNA-based network and a TCP/IP network in collaboration with the z/OS Communications Server. Application programs can use IMS Message Format Services (MFS) to define the transaction message input and output metadata and to build the input and output screen for an interactive terminal end user. MFS parses the data to manage the input and output data streams for the application program. To provide network protocol transparency, transaction message integrity and communication device-independent application programming, IMS TM includes a queue manager. Figure 3-1 shows the main interfaces to IMS TM: Open Transaction Manager Access (OTMA), IMS Connect, virtual telecommunications access method (VTAM), and advanced program-to-program communication (APPC).

IMS TM also provides two ways to couple multiple subsystems for application program communication:

- **Multiple Systems Coupling (MSC).** An IMS protocol for coupling IMS subsystems. MSC provides IMS application program workload distribution across a single system boundary or across geographical boundaries.
- **Intersystem communication (ISC).** An SNA protocol supported by IMS TM that provides access to distributed servers and distributed transaction processing between different subsystems. An application in one subsystem communicates with an application in a different subsystem. ISC is used for access to IMS from other application runtime environments and for access from IMS application programs.

To enable cooperative application processing, SNA introduced APPC, and IMS TM was enhanced to support the APPC protocols while preserving the existing DL/I API. This programming model is used for direct access to and from IMS for coordinated application program processing.

OTMA is an IMS protocol that provides an open interface to IMS TM for transparent access to IMS transaction processing application programs. An OTMA client such as IMS Connect can be in the same logical partition as OTMA, or on a different one. IMS Connect uses OTMA to support direct access with a TCP/IP-based network. The direct access programming model provides an IMS Connect client access to IMS only when IMS is available to process the request, and instant notification when the access attempt fails.

As communication technologies evolve, the IMS TM DL/I data communication interface will continue to provide the support for access to and from IMS-managed application program business services.

Figure 3-1 shows the interfaces for accessing IMS.

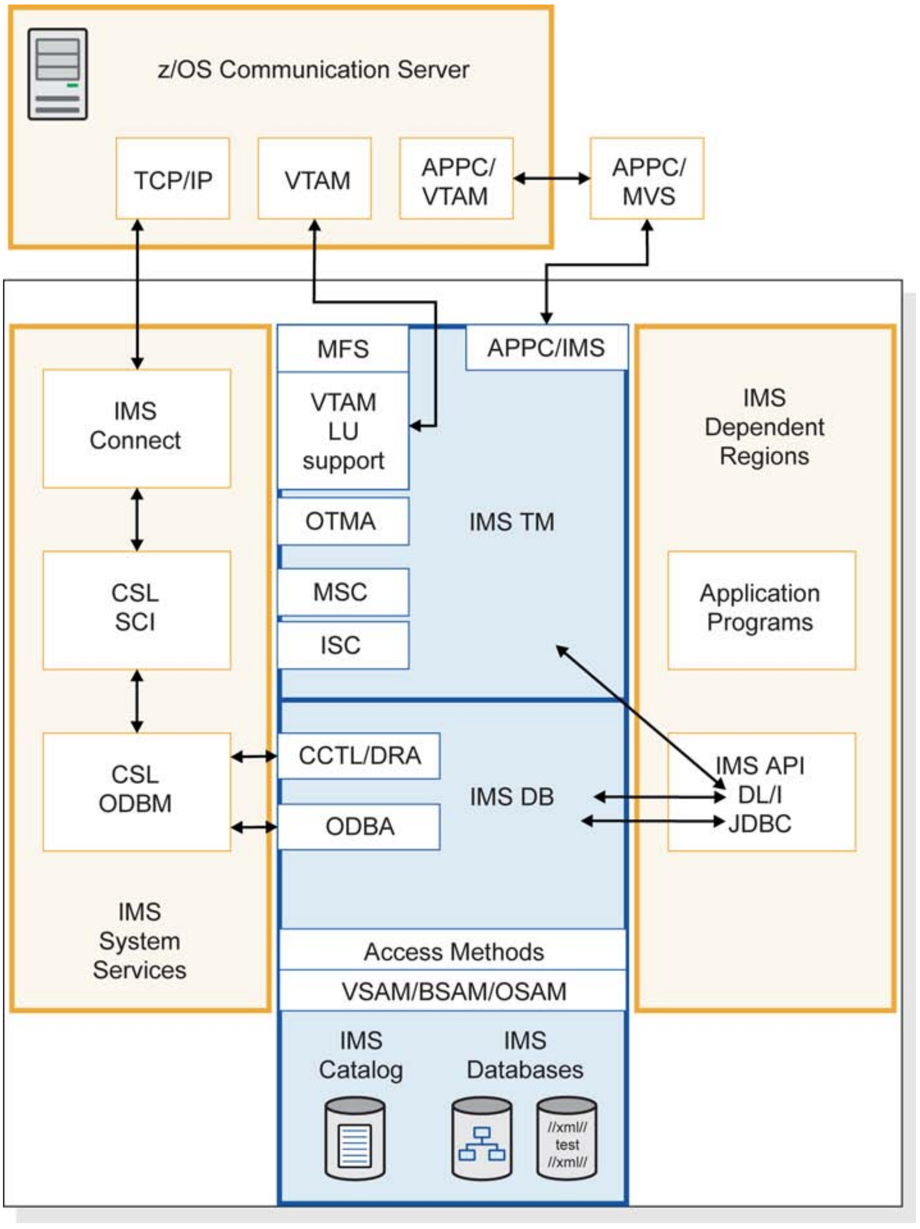


Figure 3-1 Interfaces for Accessing IMS

IMS Connect

IMS Connect is an IMS System Services function that provides high-performance TCP/IP communications between one or more IMS Connect clients and one or more IMS systems. IMS Connect supports both IMS DB and IMS TM systems.

IMS Connect provides access to IMS DB through the Open Database Manager (ODBM) component of the IMS Common Service Layer (CSL), for direct access to databases that are managed by IMS DB. IMS Connect and ODBM combined provide a Distributed Relational Database Architecture™ (DRDA) target server function for IMS database access from DRDA client applications.

IMS Connect provides access to IMS TM for transaction processing support through OTMA.

Figure 3-2 shows the connectivity options that are available with IMS Connect, ODBM, and OTMA.

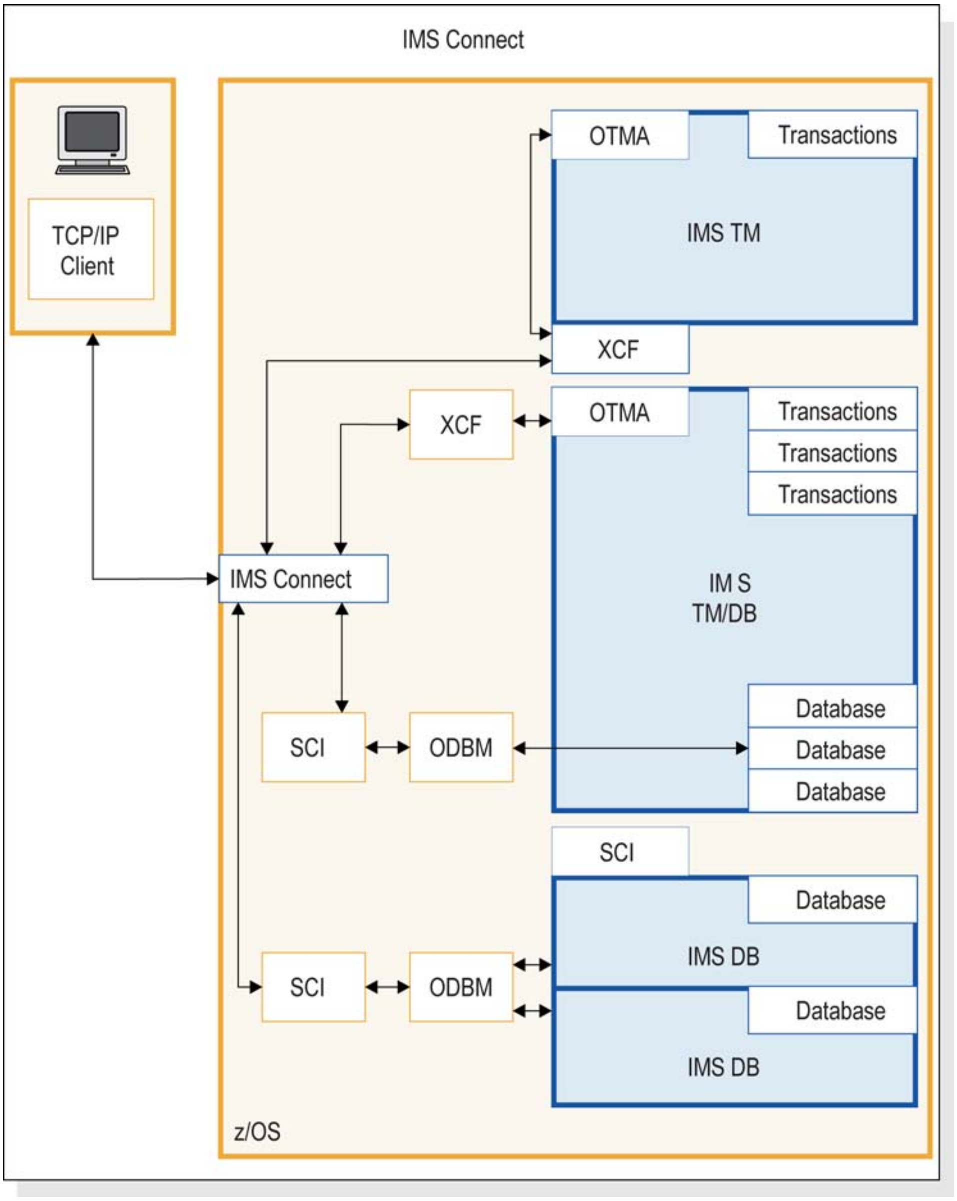


Figure 3-2 IMS Connect Access to IMS TM and IMS DB

One or more IMS Connect instances can connect to OTMA by using z/OS cross-system coupling facility (XCF) services and to ODBM by using IMS Structured Call Interface (SCI) services. One OTMA can be configured for one IMS TM system.

One or more ODBM instances can be configured to access one or more IMS DB systems.

These connectivity options provide the availability and scalability qualities of service that business entities require.

Accessing IMS DB Using JDBC

The IMS solutions for Java development provide the implementation of Java DL/I calls and the JDBC API. Figure 3-3 shows how Java application code can access IMS databases by using DL/I or JDBC, and use DL/I to access the IMS message queues from IMS dependent regions. The IMS catalog contains the IMS database metadata for access to IMS databases. For more information about writing Java applications for IMS, see Chapter 18, "Application Programming in Java."

Accessing IMS from Other Application Runtime Environments

The IMS architecture shown in Figure 3-1 shows the interfaces that are available for access to IMS from other application runtime environments.

Accessing IMS DB

Data is one of the largest assets in any business, and accelerating an organization's ability to share and deliver trusted information is essential to the success of that business. IMS has 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.

Figure 3-3 illustrates how the IMS database resource adapter (DRA) interface, the IMS Open Database Access (ODBA) interface, and Open Database Manager (ODBM) provide access to databases that are managed by IMS DB.

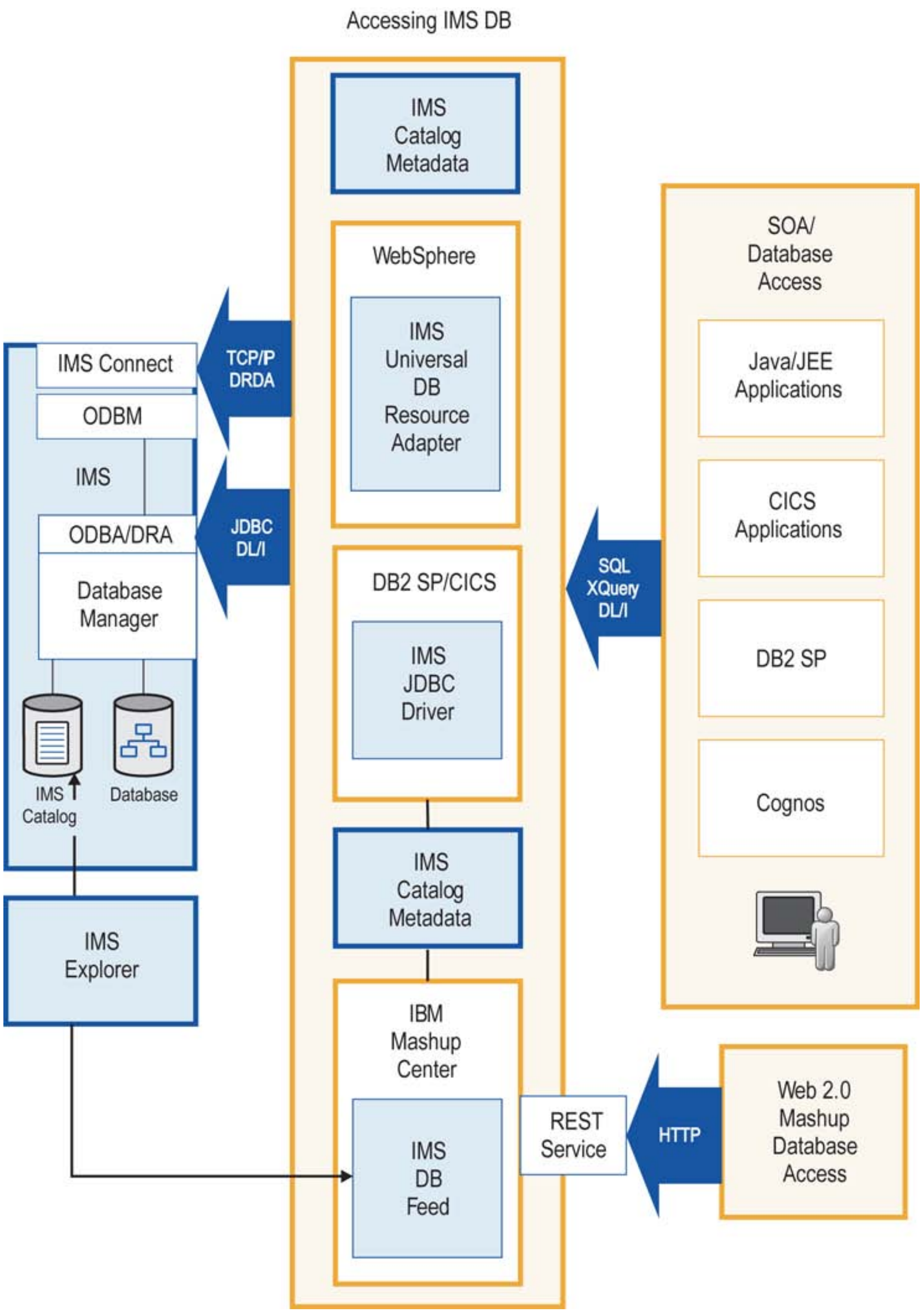


Figure 3-3 Accessing IMS DB from Other Application Runtime Environments

The coordinator controller (CCTL) DRA provides connectivity to IMS DB for application programs managed by a CCTL that is processing in the same logical partition as IMS DB. These application programs use DL/I calls or Java SQL calls to access IMS databases. The CCTL manages the coordination of changes to IMS databases. IBM CICS Transaction Server for z/OS is one of the runtime environments that support the CCTL DRA interface for CICS application program access to IMS DB.

The ODBA callable interface is available to any z/OS application program processing in the same logical partition as IMS DB. The z/OS Resource Recovery Services (RRS) function coordinates the changes to IMS databases that are made through the ODBA callable interface.

Both DB2 for z/OS stored procedures and WebSphere Application Server for z/OS Java EE applications support the ODBA interface to provide access to IMS DB.

ODBM is a CSL environment that manages distributed and local access to IMS databases. ODBM routes incoming database requests to an IMS system in the same logical partition as ODBM. An ODBM client such as IMS Connect can be in the same or a different logical partition as ODBM.

Java applications can access databases that are managed by IMS DB by using any of the following IMS Universal drivers:

- **IMS Universal DB resource adapter.** A Java EE Connector Architecture (JCA)-compliant service that Java EE applications programs use to leverage IMS databases using SQL calls
- **IMS Universal JDBC driver.** Supports access to IMS databases using SQL calls
- **IMS Universal DL/I driver.** An IMS-specific Java API for DL/I access to IMS databases using Java methods that are based on IMS DL/I semantics

Two types of connectivity are supported by the IMS Universal drivers:

Type 2. For local connectivity to IMS databases in the same logical partition

Type 4. For distributed connectivity using TCP/IP through IMS Connect and ODBM

The type-4 connection is based on the open standard DRDA specification and the distributed data management (DDM) architecture commands. ODBM translates the incoming DDM protocol database requests received from IMS Connect into DL/I calls and routes them to the IMS that is managing the requested database. ODBM translates the IMS database results to the DDM protocol before returning the output.

The use of DRDA solves the problem of obsolete data after data replication. Before this solution was available, customers had to replicate their IMS hierarchical data to a relational data store to support query tools for business intelligence. Cognos® is an example of business intelligence software that supports standard JDBC drivers and is able to use the IMS Universal JDBC driver for type-4 connectivity to access IMS databases.

Accessing IMS TM

Figure 3-4 shows how the IMS architecture supports open access to IMS TM. Customers that have IMS and CICS use ISC to leverage their transaction processing workload investment.

DB2 for z/OS provides two stored procedures for direct access to IMS TM using OTMA. The DSNAIMS stored procedure is used to send single segment input messages to the IMS transaction processing program, and the DSNAIMS2 stored procedure includes multi-segment input message support.

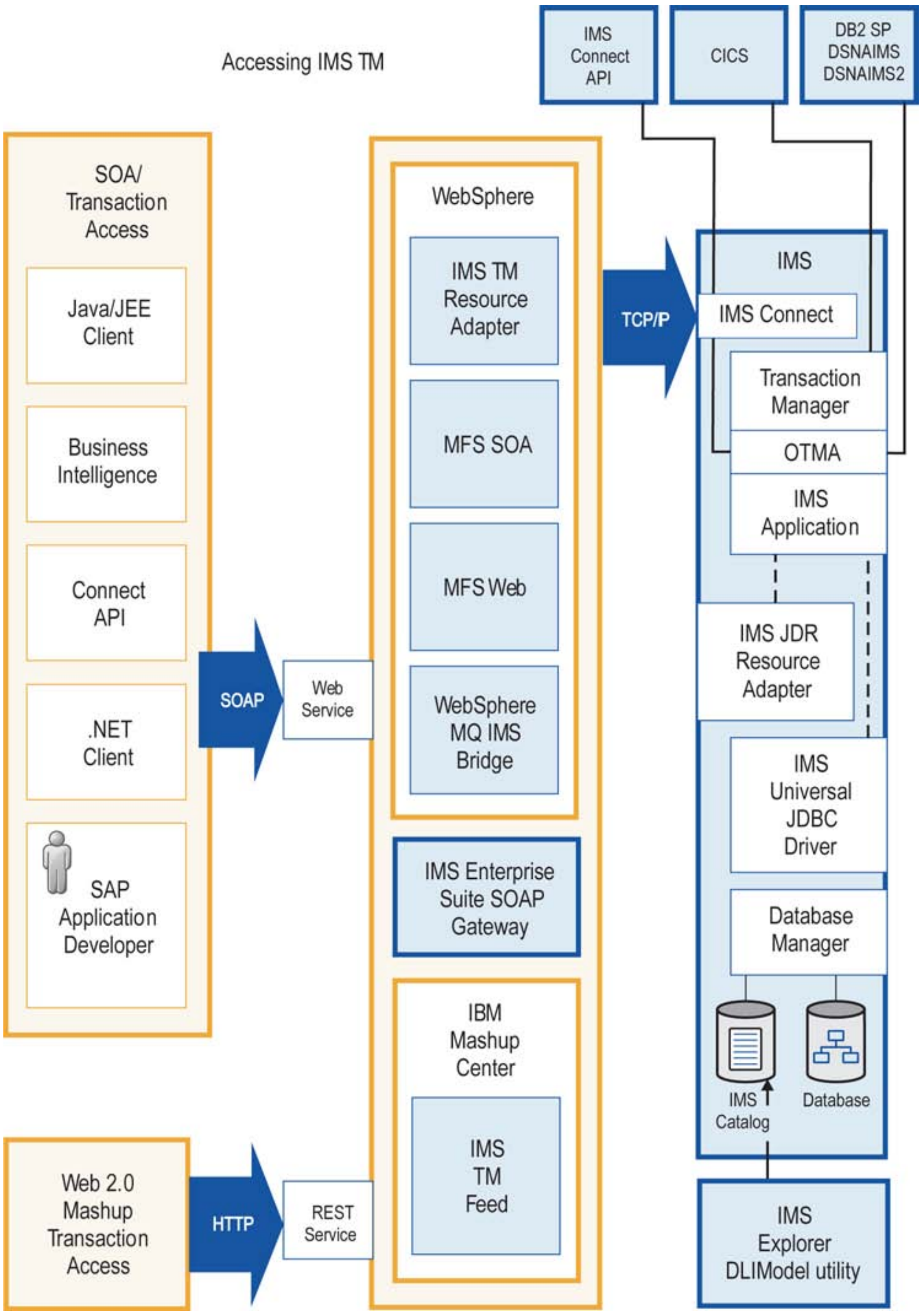


Figure 3-4 Accessing IMS TM from Other Application Runtime Environments

The WebSphere MQ-IMS bridge OTMA client component of WebSphere MQ for z/OS provides network-transparent access from WebSphere MQ client applications to IMS transaction processing application programs. The bridge is used to reengineer access controlled by

3270-connected terminals to be controlled by MQ messages, without having to rewrite, recompile, or relink IMS application programs. This programming model provides indirect communication access to IMS. The client application is assured of message delivery but does not know whether IMS is available to process the request. WebSphere MQ also provides an IMS Adapter interface for IMS application programs written to use the MQ API.

Access to and from IMS Using the IMS SOA Integration Suite Components

The IMS SOA Integration Suite middleware functions and tools (available to IMS customers at no additional cost) provide open access to the assets managed by IMS. These solutions support open integration technologies that extend access to IMS transaction processing application programs and IMS databases with minimal or no IMS application code changes required. IMS Connect provides the TCP/IP connectivity support for the IMS SOA Integration suite.

The IMS Enterprise Suite, part of the IMS SOA Integration Suite, provides components for z/OS and distributed platforms to facilitate application development, simplify IMS metadata generation, and enable IMS business event data monitoring.

IMS Enterprise Suite Connect APIs

The IMS Enterprise Suite Connect APIs provide the services that IMS Connect client applications written in Java or C/C++ use to access IMS TM. The Connect APIs facilitate the development of custom client applications running in other runtime environments that need to leverage the business logic and database access provided by IMS TM transaction processing application programs. These APIs open connections to IMS Connect on behalf of the client application, assemble messages to send to IMS Connect, including the required IMS Connect header, and manage the IMS Connect message protocol by sending and receiving the appropriate messages for interactions with IMS Connect.

The Connect API for Java supports Secure Sockets Layer (SSL) for securing TCP/IP communications between client applications and IMS Connect. Figure 3-5 shows how Web 2.0 applications developed in a WebSphere sMash environment access IMS TM by using the Connect API for Java. The IMS application program is represented as a representational state transfer (REST) service to the Web 2.0 application using the WebSphere sMash Eclipse plug-in.

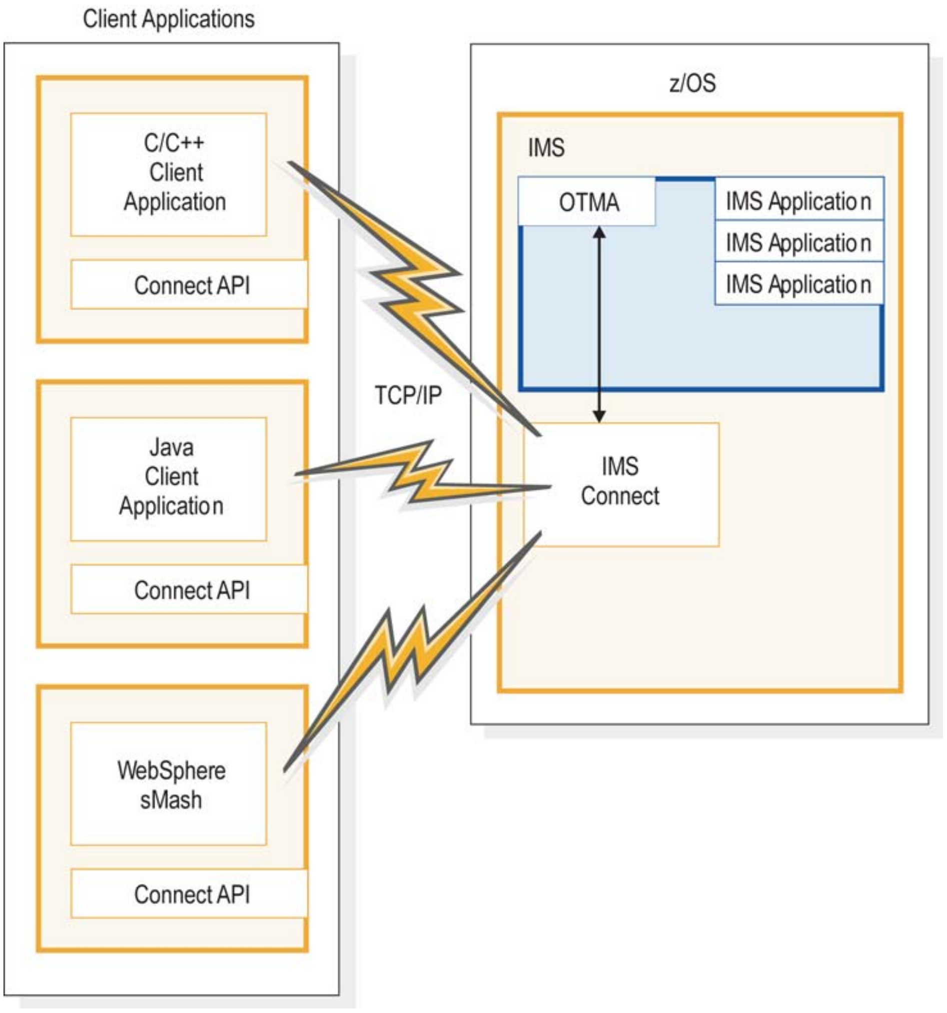


Figure 3-5 IMS Enterprise Suite Connect APIs

IMS Enterprise Suite SOAP Gateway

The IMS Enterprise Suite SOAP Gateway is a web services solution that enables IMS TM transaction processing application programs to interoperate as service providers and consumers.

IMS TM transaction processing application programs can also participate in business events processing and monitoring. Business analysts and developers can monitor IMS business activities and enhance IMS business logic with the IBM business events processing and monitoring engines.

Secured communication is supported between SOAP Gateway and the client that issues the access request for an IMS service. Secured communication is also supported between SOAP Gateway and IMS Connect. SOAP Gateway supports web services security (WS-Security), where the user identity or security token is authenticated on a per-message basis for access to IMS TM transaction processing application programs. Authentication information is passed on a per-web service basis. The SOAP Gateway solution is used when Java EE qualities of service are not required.

IBM Rational Developer for System z is an application development tool that assists in the development of traditional mainframe applications by providing the tooling to enable an IMS application as a web service provider or consumer.

For the IMS application web service provider scenario, IMS COBOL or PL/I application input and output message metadata is used to generate a Web Services Description Language (WSDL) file. The generated WSDL file describes the functions provided by the IMS application, and how the input and output messages are structured in order to invoke the function.

For the IMS application web service consumer scenario, the WSDL file of the web service to be called is imported into Rational Developer for System z to generate the required artifacts.

Figure 3-6 shows the IMS service provider flow: The SOAP Gateway server performs the conversion of SOAP messages and the IMS Connect runtime supports the XML transformation to provide device independence for IMS transaction processing application programs.

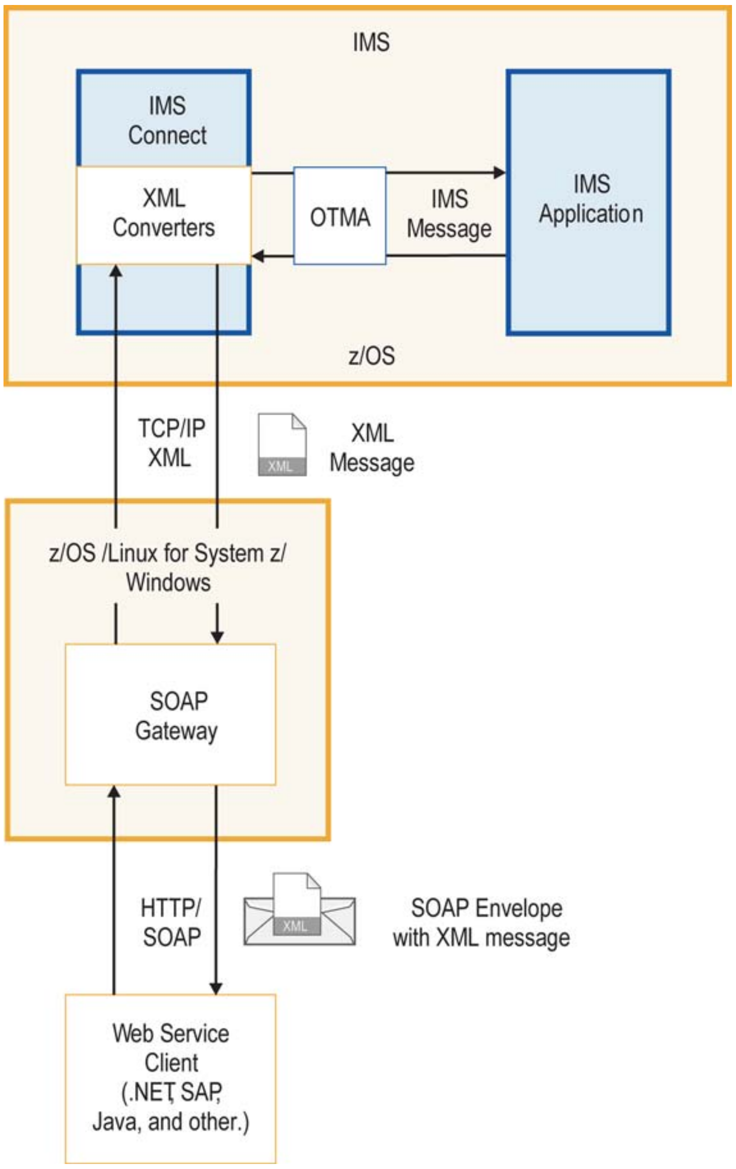

IMS
IMS Connect
XML Converters
OTMA
IMS Message
IMS Application
z/OS
TCP/IP XML
XML
XML Message
z/OS /Linux for System z/ Windows
SOAP Gateway
HTTP/ SOAP
XML
SOAP Envelope with XML message
Web Service Client (.NET, SAP, Java, and other.)

Figure 3-6 IMS Enterprise Suite SOAP Gateway

SOAP Gateway customer example: An equipment manufacturing company needed to send status information from its factory devices to IMS. The customer used SOAP Gateway on System z to provide direct access from its .NET applications to its IMS transaction processing application programs.

IMS TM Resource Adapter

The IMS TM resource adapter is used by Java applications, Java EE applications, or web services to access IMS transaction processing application programs. IMS applications that run in IMS-dependent regions can use the IMS TM resource adapter to make synchronous or asynchronous callout requests to external Java EE applications.

When a Java application or a web service issues a request to access an IMS transaction processing application program, the IMS TM resource adapter communicates with IMS Connect through TCP/IP. IMS Connect then sends the transaction request to IMS via OTMA, and the IMS transaction processing application programs is activated. The response is returned to the Java application using the same communication path.

When an IMS-managed application program issues a callout request using IMS TM services, the IMS TM resource adapter polls IMS Connect to retrieve the callout request. The Java application processes the request and returns any response data to IMS. The response is returned in the same transaction scope for a synchronous callout request, and to a different transaction processing instance for an asynchronous callout request, as shown later in Figure 3-9.

The IMS TM resource adapter is used when a Java application requires qualities of service that are provided by a Java EE application server, such as connection management, transaction management, and security management. Figure 3-7 shows the WebSphere Application Server runtime environments where IMS TM resource adapter is deployed.

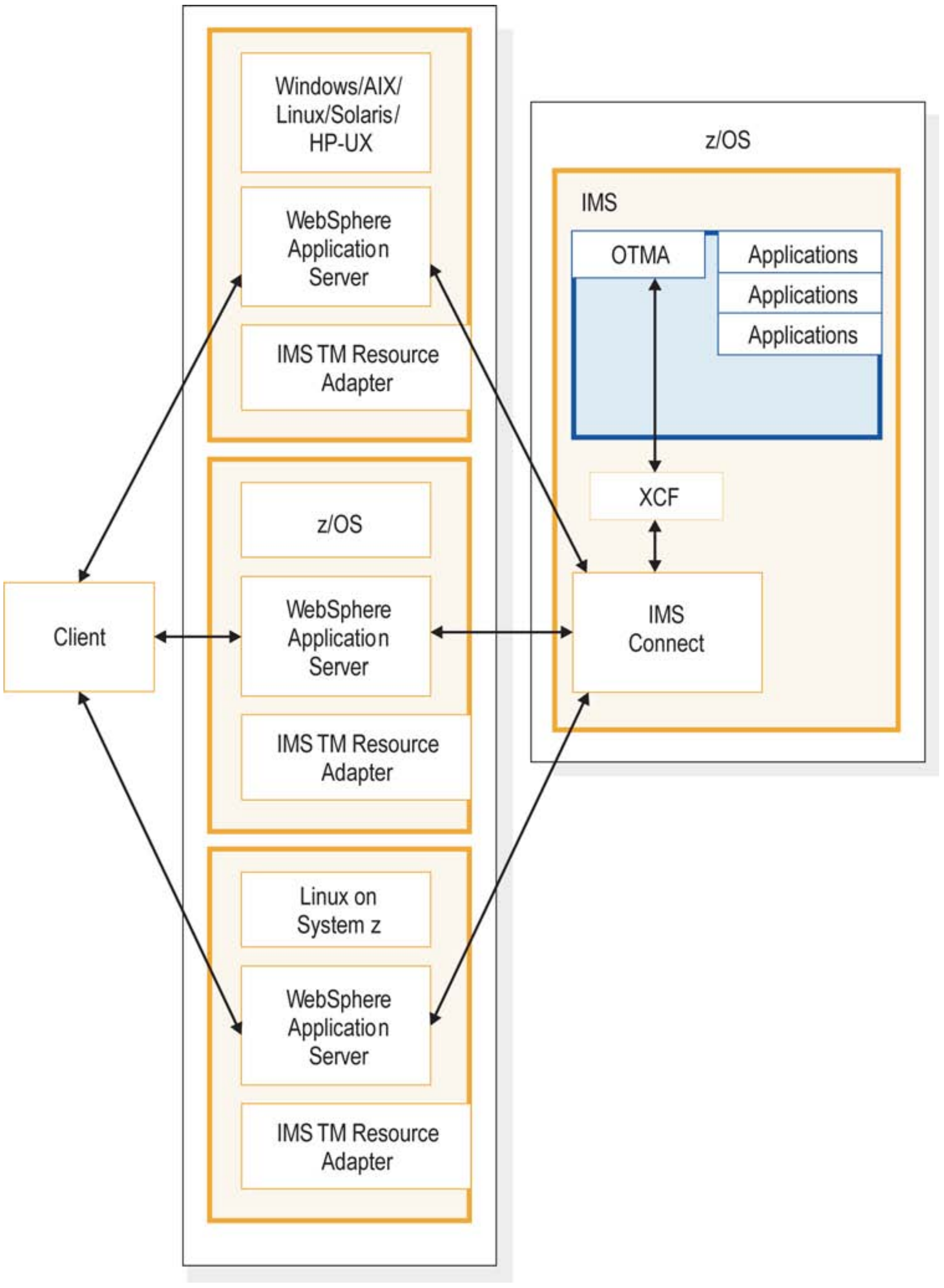


Figure 3-7 IMS TM Resource Adapter

The IMS application program expects the input and output message data to be in its language structure. Rational Application Developer for System z generates language format handlers based on the IMS COBOL or PL/I application input and output message metadata. The IMS TM resource adapter uses the format handlers to serialize values from Java to an input data buffer that the IMS application program expects, and to convert the response data buffer into Java values.

IMS MFS Web Enablement

IMS MFS provides the transaction input and output message metadata used to access MFS-based IMS transaction processing application programs. The use of the MFS metadata enables the reuse of existing MFS-based IMS business logic without modification of the application program source.

IMS MFS Web Enablement provides the tooling utility and runtime support for a web-based interface. The tooling utility is comprised of the MFS XML Utility and the MFS Importer. The MFS XML Utility invokes the MFS Importer to parse MFS source message input and output metadata and generates corresponding XML Metadata Interchange (XMI) files.

The runtime support that is deployed to a WebSphere Application Server includes a Java servlet to dynamically render MFS-based web pages on browsers by using cascading style sheets (CSSs), and an adapter that loads the metadata generated by the MFS XML Utility and communicates with IMS Connect and IMS through the IMS TM resource adapter.

This solution is used to transition from a 3270 emulator-based interface to a standard browser interface.

MFS SOA Support

MFS SOA support enables the reuse of existing MFS-based IMS business logic as web services, web pages, Enterprise JavaBeans components, or integrated into business processes. The MFS SOA development component is a part of the Enterprise Metadata Discovery framework in Rational Application Developer. The development component is used to generate J2C Java beans and J2C Java Data Binding classes based on MFS source message input and output metadata. The runtime component is deployed to a WebSphere Application Server and uses the IMS TM resource adapter to communicate with IMS Connect and IMS.

MFS SOA customer example: An electronics distributor needed integrated access to its distributed platforms and its IMS MFS-based applications. The customer used IMS MFS SOA support to integrate its IMS assets and its distributed platform assets into a Web 2.0 mashup solution.

MFS SOA Business Process Execution Language (BPEL) supports the integration of MFS-based IMS transaction processing application programs as components (services) to create a business process. The MFS SOA development component is a part of the Enterprise Metadata Discovery framework in the IBM Integration Designer. The MFS source message input and output metadata is used by the Integration Designer External Service Discovery framework to generate an Enterprise Information System binding import based on the principles of Service Component Architecture (SCA). SCA is a model that provides a means for expressing business logic and business data as SOA services, regardless of implementation details. The MFS source message input and output metadata is used by the Integration Designer to generate the components and services that are deployed to an IBM Process Server, which uses the IMS TM resource adapter to communicate with IMS Connect and IMS.

IMS Solutions for Java Development

IMS solutions for Java development provide the support for the Java programming language. You can write Java applications to access IMS databases and process IMS transactions.

IMS Enterprise Suite DLIModel Utility Plug-In

The IMS Enterprise Suite DLIModel utility plug-in uses IMS database information to generate application-independent metadata for Java application programming.

IMS Enterprise Suite Explorer for Development

The IMS Enterprise Suite Explorer for Development (IMS Explorer) is an Eclipse-based graphical tool that simplifies IMS application development tasks. The tool can be used by IMS application developers or database administrators to do the following tasks:

- Display and edit IMS database and program definitions
- Display a relational view of IMS data
- View and edit z/OS data sets and z/OS UNIX® files
- Submit JCL, and view output and job logs
- Build SQL statements to process IMS data

Figure 3-8 shows the IMS Explorer graphical user interface (GUI).

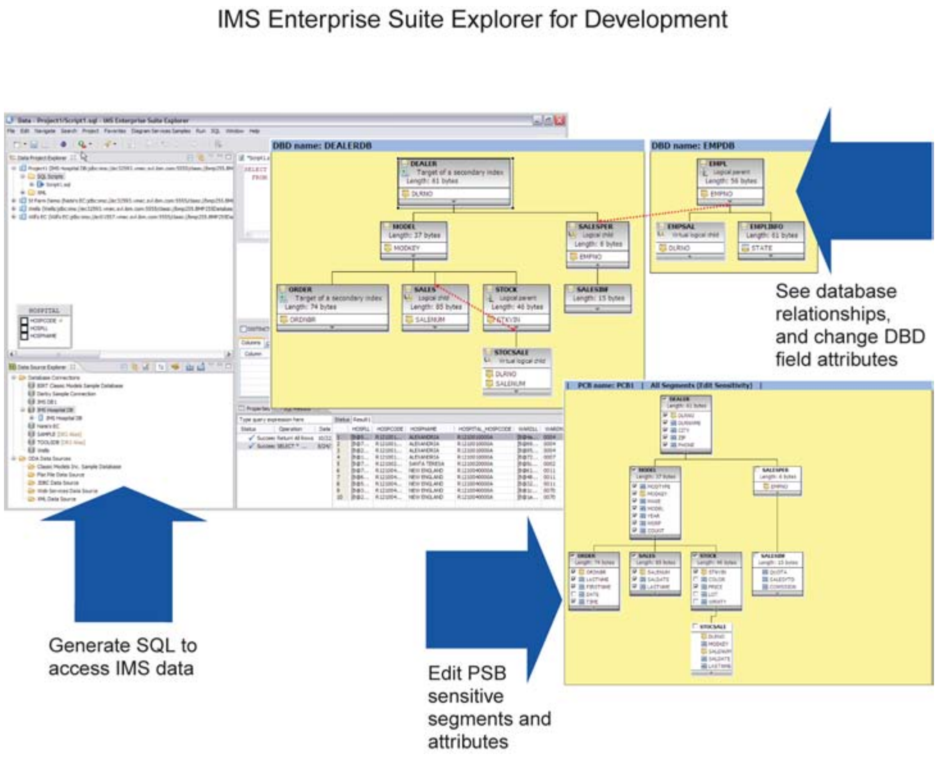

IMS Enterprise Suite Explorer for Development
DBD name: DEALERDB
DBD name: EMPDB
See database relationships, and change DBD field attributes
Generate SQL to access IMS data
Edit PSB sensitive segments and attributes

Figure 3-8 IMS Explorer GUI

IMS Explorer provides a modern interface to display and edit IMS database and program definitions. The IMS Explorer Import wizard can import the database and program definitions from a local file system or from z/OS. IMS database definition information about data field layouts can be obtained by importing one or more data structure files (COBOL copybooks or PL/I includes) into your IMS Explorer project. When you save changes to the database and program definitions, IMS Explorer generates the database and program definition source files with the changes and the metadata to update the IMS catalog.

The IMS catalog provides a validated and comprehensive view of IMS database metadata and is fully managed by IMS. The metadata is used to connect to an IMS database from the IMS Explorer Data Source Explorer function or to develop Java applications for IMS. This capability improves programmer productivity and helps reduce IMS application development costs.

IMS XML DB

IMS DB supports data type neutrality, which enables the storage of a variety of data formats. This flexibility can be leveraged as new data paradigms emerge. IMS XML DB is a Java application that uses the IMS Universal JDBC driver to store and retrieve XML data. The Java application can be deployed to IMS, CICS, DB2, or WebSphere Application Server runtime environments. The IMS Universal JDBC driver is able to decompose the XML document and update the IMS database. IMS application programs that access IMS databases using the DL/I API can retrieve the updated data. The IMS universal JDBC driver can also retrieve existing data from an IMS database and compose the data into an XML document that can be sent to the client. Because IMS DB supports data transparency, IMS XML DB can also be used to store XML documents intact.

IMS Web 2.0 Solutions for IMS DB and IMS TM

Web 2.0 applications provide cooperative Internet services, integrate data, and leverage business processes by using XML data feeds. IMS Web 2.0 solutions provide the ability to integrate IMS TM-managed transaction processing application programs and IMS DB-managed databases with Web 2.0 applications.

The IMS Web 2.0 solution for creating XML data feeds to access IMS-managed transaction processing application programs and databases is embedded in the IBM Mashup Center. The IBM Mashup Center has a graphical user interface for creating and customizing feeds and for mashing up data from various sources into a dynamic application with a single view of information.

The IMS transaction feed generator is an IBM Mashup Center plug-in that handles the communication with IMS Connect for feeds to access IMS transaction processing application programs. The feed is created from the input and output message structures of the IMS transaction processing application program. When WebSphere Application Server receives an HTTP request for an IMS transaction feed from a web client, it passes the request to IBM Mashup Center, which invokes the IMS transaction feed generator. The IMS transaction feed generator processes the parameters, establishes a connection with IMS Connect, and sends the request as an XML message. The XML adapter function of IMS Connect converts the XML message and uses OTMA to send the data as an input message for the IMS transaction processing application program. The response output message is converted by IMS Connect into XML response data. The response is then returned to the IMS transaction feed generator, which converts the response into a data feed format and sends it to the client as an HTTP response. The IMS Connect runtime XML transformation provides device independence for IMS transaction processing application programs.

For feeds that access IMS databases, the IMS Universal DB resource adapter is used to communicate between IMS Connect and WebSphere Application Server. The feed is created from an IMS database based on the database view metadata. When the IBM Mashup Center receives an HTTP request for an IMS database feed from a web client, it passes the request to the enterprise database feed generator. The enterprise database feed generator constructs the SQL query and sends it to the IMS universal DB resource adapter that is deployed in WebSphere Application Server. The IMS universal DB resource adapter communicates with IMS through IMS Connect, which uses ODBM to access the IMS database.

You can also develop Web 2.0 applications in WebSphere sMash to access IMS transaction processing application programs. WebSphere sMash is a web application platform and environment for developing and running web applications that are based on Web 2.0 technologies. WebSphere sMash applications can be written in Java, Groovy, or PHP. The IMS Enterprise Suite Connect API for Java provides the connectivity access to IMS.

Accessing from IMS

Accessing from IMS is accomplished by IMS application programs using IMS asynchronous and synchronous callout capabilities to communicate with application programs managed by IMS, and other application runtime environments.

Asynchronous callout is used when the IMS application program does not expect a response, or when the response message will be processed by another IMS application program instance. IMS application programs use the DL/I ISRT call to the IMS-managed ALTPCB to send asynchronous messages.

Synchronous callout is used when the IMS application program needs the response in the same transaction scope. IMS application programs use the DL/I ICAL call to send and receive messages for synchronous callout processing. Java application programs for IMS use the IMS Enterprise Suite Java Message Service (JMS) API for synchronous callout processing.

Figure 3-9 shows how IMS TM services and some IMS SOA Integration Suite components support IMS as a service consumer. The use of these capabilities is described in Part IV, "IMS Application Development."

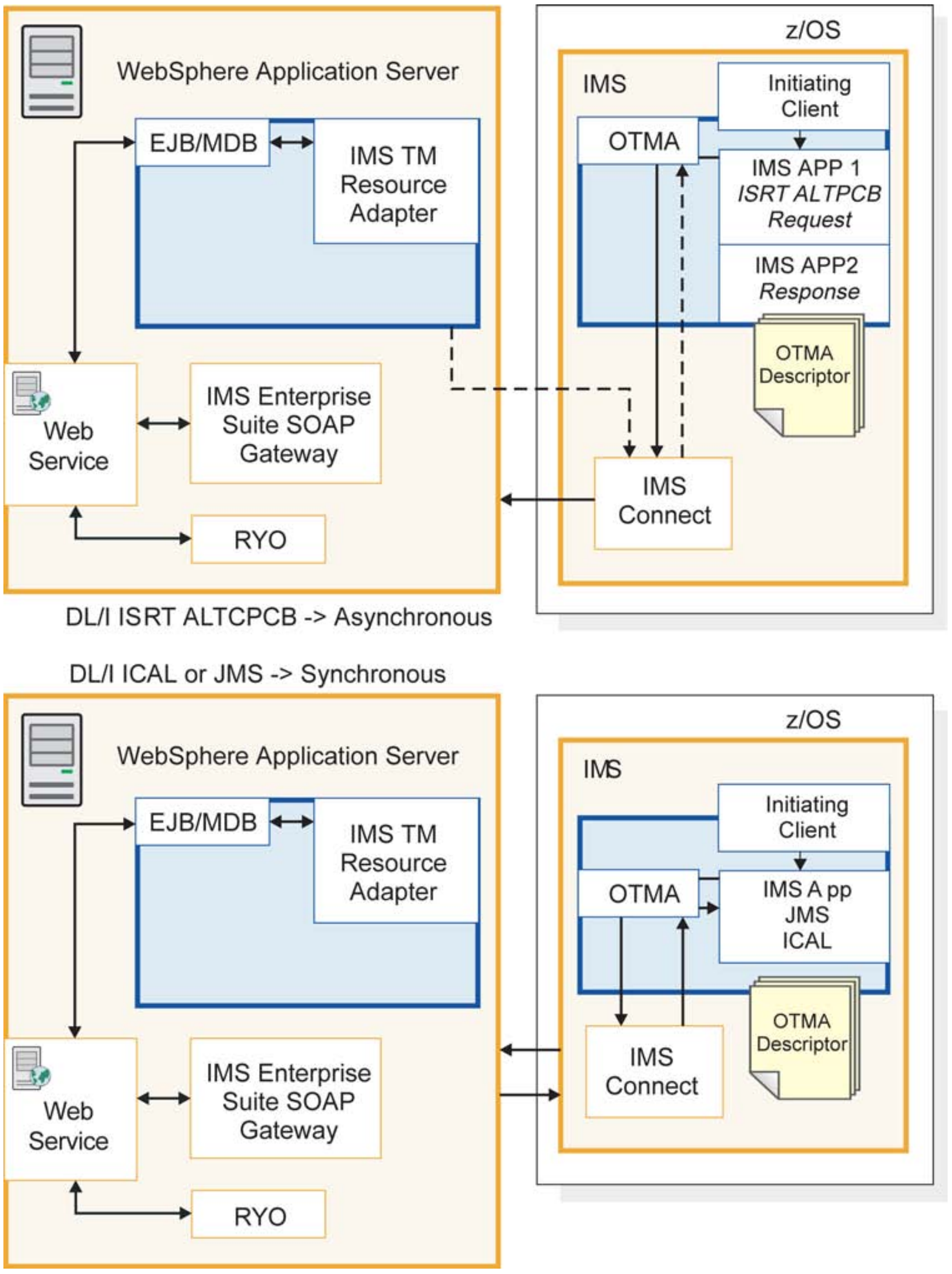


Figure 3-9 Access from IMS Applications: IMS as a Service Consumer

IMS application programs can use z/OS supported communication interface calls for asynchronous or synchronous callout requests. For TCP/IP sockets, the z/OS Communications Server provides a standard sockets API to support C and Java application programs. The extended sockets API is available for Assembler, COBOL, and PL/I application programs.

APPC on z/OS supports the open standard Common Programming Interface for Communication API for direct communication to APPC-based application programs. The WebSphere MQ Client API can be used to send and receive MQ-managed messages.

The direct use of these communication solutions by IMS application programs is not managed by IMS. Because IMS does not participate in support of the callout request, any error-recovery processing must be provided by the IMS application program.

Accessing to and from IMS

IMS continues to innovate by providing functions that not only meet today's information technology and business requirements, but also prepare organizations for the future. Instead of rip-and-replace, organizations can count on IMS to protect and further extend the value of their existing IMS assets through technological innovations and evolution.

CHAPTER 4

IMS and z/OS

IMS subsystems are implemented on a z/OS system. IMS uses some of the facilities of the z/OS operating system.

In This Chapter

- How IMS Relates to z/OS
- Structure of IMS Subsystems
- Running an IMS System
- Running Multiple IMS Systems
- How IMS Uses z/OS Services

How IMS Relates to z/OS

IMS is a large application that runs on z/OS. IMS and z/OS are both designed to provide the most efficient use of the hardware and software components.

IMS runs as a z/OS subsystem and uses several address spaces: one controlling address space, several separate address spaces that provide IMS services, and several address spaces that run IMS application programs. z/OS address spaces are sometimes called *regions*, as in the *IMS control region*. The term *region* is synonymous with a z/OS address space.

Structure of IMS Subsystems

This section describes the various types of z/OS address spaces used by IMS, and their interrelationships.

The IMS control region is the core of an IMS subsystem, running in one z/OS address space. Each control region uses many other address spaces that provide additional services to the control region, and to the address spaces in which the IMS application programs run.

Some IMS applications and utilities run in separate, stand-alone regions, called *batch regions*. Batch regions are separate from an IMS subsystem and its control region, and have no connection with it.

IMS Control Region

The IMS control region is a z/OS address space that can run as a started task, initiated through a z/OS `START` command, or as a job, initiated by submitting job control language (JCL). (JCL is a control language that is used to identify a job to an operating system and to describe the job's requirements.)

The IMS control region provides the central point of control for an IMS subsystem:

- Provides the interface to z/OS for the operation of the IMS subsystem
- Controls, schedules, and dispatches the application programs that are running in separate regions, called *dependent regions*
- Provides the interface to the Systems Network Architecture (SNA) network for IMS TM functions
- Provides the Open Transaction Manager Access (OTMA) interface for access to non-SNA networks, other subsystems such as WebSphere MQ, and other applications exchanging messages with IMS
- Provides the Open Database Access (ODBA) interface for IMS Open Database Manager (ODBM), DB2 for z/OS stored procedures, and other z/OS application programs

The IMS control region also provides all logging, restart, and recovery functions for the IMS subsystems. The virtual telecommunications access method (VTAM®) terminals, message queues, and logs are all attached to this region. Fast Path (one of the IMS database types) database data sets are also allocated by the IMS control region.

A z/OS type-2 supervisor call (SVC) routine is used for switching control information, messages, and database data between the control region, all other regions, and back.

Four different types of IMS control regions can be defined using the IMS system definition process. You choose the control region type based on which IMS functions you will use. The four types of IMS control regions support the four IMS environments.

IMS Environments

Each of the IMS environments is a distinct combination of programs that supports distinct processing goals. In conjunction with the IMS External Subsystem Attach Facility (ESAF), these environments can connect to external subsystems such as DB2 and WebSphere MQ. This interface is known as the External Subsystem (ESS) interface. The four IMS environments are as follows:

- DB/DC, which contains all the functionality of both IMS TM and IMS DB (see the section “IMS DB/DC Environment”).
- DBCTL (pronounced DB Control), which contains the functionality of only IMS DB (see the section “IMS DBCTL Environment”).
- DCCTL (pronounced DC Control), which contains the functionality of only IMS TM (see the section “IMS DCCTL Environment”).
- Batch, which contains the functionality of IMS DB, but is used only for batch jobs (see the section “IMS Batch Environment”).

IMS DB/DC Environment

The DB/DC environment has both IMS TM and IMS DB installed and has the functionality of the entire IMS product.

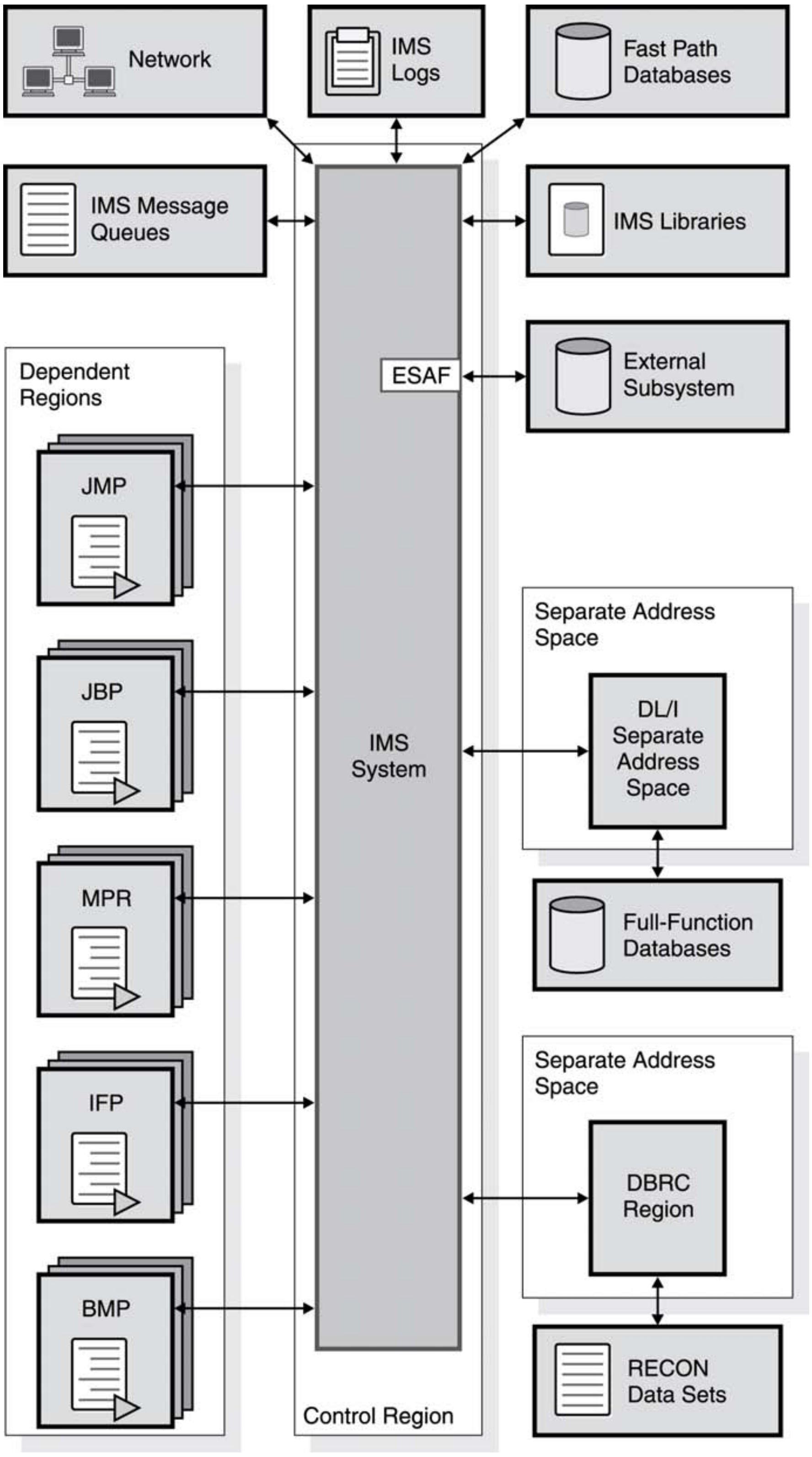

Network
IMS Logs
Fast Path Databases
IMS Message Queues
IMS Libraries
ESAF
External Subsystem
Dependent Regions
JMP
JBP
MPR
IFP
BMP
IMS System
Separate Address Space
DL/I Separate Address Space
Full-Function Databases
Separate Address Space
DBRC Region
RECON Data Sets
Control Region

Figure 4-1 Structure of a Sample IMS DB/DC Environment

As shown in Figure 4-1 the DB/DC control region provides access to the following:

- Network, which might include a z/OS console, terminals, web servers, and more
- IMS message queues
- IMS logs
- Fast Path databases
- Data Language/Interface (DL/I) separate address space
- Database Recovery Control (DBRC) facility region
- Message processing regions (MPRs)
- IMS Fast Path (IFP) regions
- Java message processing (JMP) regions
- Java batch processing (JBP) regions
- Batch message processing (BMP) regions

IMS DBCTL Environment

The DBCTL environment has only IMS DB installed. DBCTL can provide IMS database functions to non-message-driven batch message programs (BMP and JBP application programs) connected to the IMS control region, and to application transactions running in CICS regions, as shown in Figure 4-2.

The IMS Open Database Manager (ODBM) can also be used in a DBCTL environment.

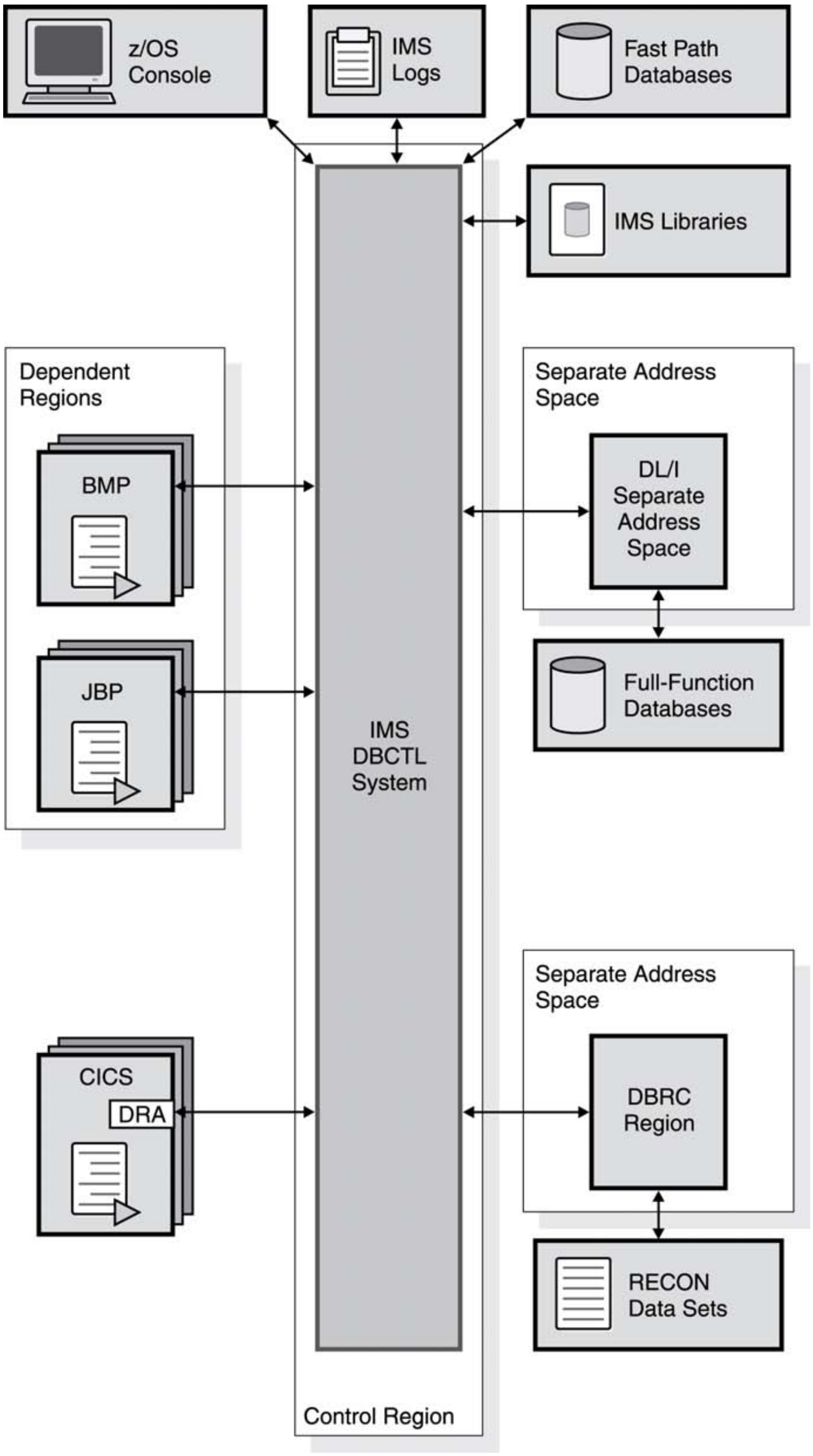

z/OS Console
IMS Logs
Fast Path Databases
IMS Libraries
Dependent Regions
BMP
JBP
CICS
DRA
IMS DBCTL System
Control Region
Separate Address Space
DL/I Separate Address Space
Full-Function Databases
Separate Address Space
DBRC Region
RECON Data Sets

Figure 4-2 Structure of a Sample IMS DBCTL Environment

When a Customer Information Control System (CICS) connects to IMS using the database resource adapter (DRA), each CICS system has a predefined number of connections with IMS. Each of these connections is called a *thread*. Although threads are not jobs from the perspective of z/OS, each thread appears to the IMS system to be another IMS dependent region. The DL/I processing runs in one of these dependent regions when a CICS application issues a DL/I call to IMS.

When a DB/DC environment is providing access to IMS databases for a CICS region, it is referred to in some documentation as providing DBCTL services, although it might, in fact, be a full DB/DC environment and not only a DBCTL environment.

IMS DCCTL Environment

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

As shown in Figure 4-3, the DCCTL system, in conjunction with the IMS ESAF, provides a transaction manager facility to external subsystems (for example, DB2 for z/OS).

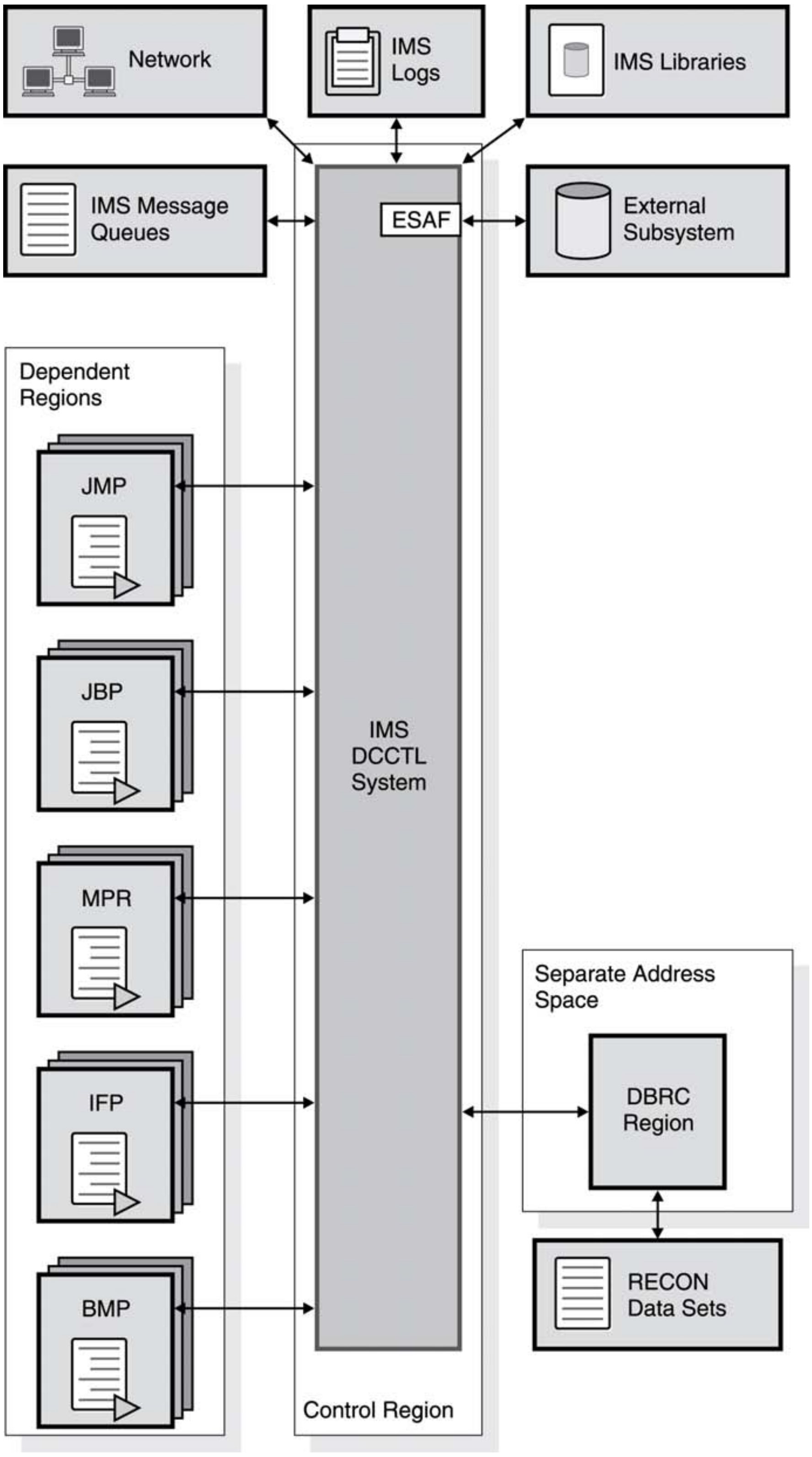

Network
IMS Logs
IMS Libraries
IMS Message Queues
ESAF
External Subsystem
Dependent Regions
JMP
JBP
MPR
IFP
BMP
IMS DCCTL System
Separate Address Space
DBRC Region
RECON Data Sets
Control Region

Figure 4-3 Structure of a Sample IMS DCCTL Environment

In a DCCTL environment, transaction processing and terminal management is identical to that in a DB/DC environment.

IMS Batch Environment

The IMS batch environment consists of a batch region (a single address space) where an application program and IMS routines reside. The batch job that runs the batch environment is initiated with JCL, like any operating-system job.

There are two types of IMS batch environments: DB batch and TM batch.

DB Batch Environment

In the DB batch environment, IMS application programs that use only IMS DB functions can be run in a separate z/OS address space that is not connected to an IMS online control region. These batch applications are typically long-running jobs that perform large numbers of database accesses. DB batch applications can access only full-function databases (which are described in Chapter 6, “Overview of the IMS Database Manager”), not Fast Path databases. They can access the full-function databases that are online to an IMS control region by using IMS database data sharing and the IRLM.

A DB batch environment is initiated by submitting a job through JCL. All the IMS code that is used by the application resides in the address space in which the application is running. The job executes an IMS batch region controller that then loads a program controller, which calls the application. Figure 4-4 shows an IMS batch region.