

FOCUS for S/390

IMS/DB Interface Users Manual and Installation Guide
Version 7.0

Contents

- 1. Introduction 1-1
 - 1.1 Environments 1-3
 - 1.2 Supported IMS Access Methods 1-3
 - 1.3 Ease of Use 1-4
 - 1.4 Efficiency 1-5
 - 1.5 Security 1-5
- 2. IMS Overview and Mapping Concepts 2-1
 - 2.1 Overview of IMS Concepts 2-1
 - 2.1.1 Hierarchical Structure 2-1
 - Sequence Fields 2-2
 - Secondary Indexes 2-3
 - Logical Relations 2-3
 - Search Fields 2-3
 - Symbolic Pointers 2-3
 - 2.1.2 Overview of IMS Access Methods 2-4
 - 2.1.3 Overview of IMS Control Blocks 2-5
 - Describing a Database: The DBD 2-5
 - Defining an Application's Access to Databases: The PSB 2-6
 - Describing a Database View and Communicating With the DBMS: The PCB 2-6
 - Key Sensitivity 2-7
 - Field Level Sensitivity 2-7
 - Status Codes 2-8
 - Key Feedback Area 2-8
 - The ACB 2-8
 - 2.1.4 Overview of DL/I Calls 2-8
 - Segment Search Arguments 2-9
 - Get Calls 2-9
 - 2.2 Mapping IMS Elements to FOCUS 2-10
 - 2.2.1 Describing the PSB: The FOCPSB 2-10
 - 2.2.2 Describing the Database: The Master File 2-11
 - Identifying the IMS Database 2-12
 - Describing IMS Segments to FOCUS 2-13
 - Describing IMS Fields to FOCUS 2-14
 - Describing IMS Segments That Have Multiple Definitions to FOCUS 2-17
 - Supporting Logical Segment Types 2-19
 - Describing Variable Length Segments to FOCUS 2-20
 - Describing a Secondary Index to FOCUS 2-21

Contents

- 2.3 Mapping IMS and FOCUS Relationships2-24
 - IMS Logical Relationships2-24
 - Alternating Between Databases.....2-24
 - The Dynamic JOIN.....2-24
- 3. Creating FOCUS Descriptions3-1
 - 3.1 The FOCPSB.....3-2
 - 3.1.1 Required FOCPSB Attributes.....3-3
 - The Header Record.....3-3
 - The PCB Record.....3-4
 - 3.1.2 Extended FOCPSB Attributes3-5
 - Describing a Partition in the FOCPSB3-6
 - Describing a Concatenated PCB in the FOCPSB.....3-7
 - 3.1.3 The FOCPSB Dataset3-8
 - 3.1.4 Dynamic PCB Selection Exit3-9
 - How the Use the IMDYNPCB Exit.....3-9
 - IMDYNPCB Exit Sample3-10
 - 3.2 The Master File3-11
 - 3.2.1 File Attributes3-12
 - 3.2.2 Segment Attributes3-12
 - SEGNAME.....3-13
 - SEGTYPE3-13
 - PARENT3-14
 - 3.2.3 Field Attributes.....3-14
 - FIELDNAME.....3-16
 - ALIAS3-17
 - USAGE.....3-18
 - ACTUAL.....3-18
 - 3.2.4 Group Fields3-19
 - 3.2.5 Using a Secondary Index.....3-22
 - 3.2.6 Segment Redefinition: The RECTYPE Attribute.....3-25
 - Accepting Multiple RECTYPE Values3-27
 - 3.2.7 Variable Length Segments: The OCCURS Segment3-27
 - The ORDER Field.....3-30
 - Example: OCCURS=n3-31
 - Example: OCCURS=fieldname.....3-32
 - Example: OCCURS=VARIABLE3-33
 - 3.3 The Access File3-34
- 4. Reporting Efficiencies.....4-1
 - 4.1 Interface Optimization.....4-2
 - 4.2 DL/I Calls.....4-3
 - 4.3 Record Selection Tests4-4

4.3.1	Access Method Restrictions	4-6
4.3.2	Rules for Constructing SSAs From FOCUS IF Tests	4-6
4.3.3	Complex Screening Conditions.....	4-8
	The SSA Buffer	4-8
	Constructing a Single SSA	4-9
	Constructing Multiple SSAs.....	4-10
	Sequentially Accessed Root Segments.....	4-13
4.3.4	WHERE Tests	4-13
4.3.5	Partial Key and Multi-Segment Requests.....	4-15
	Selection on a Partial Key	4-15
	Multi-Segment Requests	4-16
4.3.6	Auto Index Selection	4-18
4.3.7	Search Limits.....	4-20
4.4	The Dynamic JOIN Command.....	4-22
4.4.1	Single-Field Dynamic JOIN	4-23
4.4.2	Multi-Field Dynamic JOIN	4-25
4.4.3	Join Optimization	4-27
4.5	Retrieval of Unique Segments.....	4-29
4.6	JOINs With Selection Criteria.....	4-31
5.	Environments	5-1
	Fast Path Considerations	5-2
	The PSB PROCOPT to Use With the Interface	5-3
5.1	Access to IMS Through DBCTL	5-3
5.1.1	Advantages of DBCTL.....	5-4
5.1.2	Invoking the Interface in the DBCTL Environment.....	5-5
5.1.3	Implementing DBCTL.....	5-8
5.2	Access to IMS Through the XMI Server	5-9
5.2.1	Initiating the XMI Server in BMP Mode: PARM='BMP,XMI,psbname'	5-11
5.2.2	Initiating the XMI Server in DLI Mode: PARM='DLI,XMI,psbname'	5-13
5.2.3	Initiating the XMI Server With Databases Under Control of CICS	5-16
	Changing Addressability Mode of the XMI Server	5-16
	Program DFHDRP	5-17
5.2.4	Invoking the Interface in the XMI Server Environment.....	5-19
5.2.5	Additional Instructions for Using the XMI Server	5-20
	The Communication File.....	5-20
	PSBs for the XMI Server.....	5-22
	Terminating an XMI Server	5-23
5.3	Access to IMS With FOCUS Loaded by the Region Controller.....	5-24
5.3.1	Invoking the Interface in BMP Mode: PARM='BMP,FOCUS,psbname'	5-25
5.3.2	Invoking the Interface in DLI Mode: PARM='DLI,FOCUS,psbname'	5-28
5.4	Summary Chart	5-31

Contents

- 5.5 Environment Switching.....5-33
 - 5.5.1 Switching to DBCTL From the XMI Server.....5-33
 - 5.5.2 Switching to the XMI Server From DBCTL.....5-34
 - 5.5.3 Configuring the IMS Interface to Support Non-stop Processing.....5-34
- 6. Security6-1
 - 6.1 DBCTL Security6-2
 - 6.2 XMI Server Security6-3
 - 6.2.1 How Does the Exit Work6-3
 - Example: Exit6-4
 - 6.2.2 Installation of IMSECHK Exit6-6
 - 6.2.3 Tracing the Exit6-6
- A. Sample File DescriptionsA-1
 - A.1 DI21PART SampleA-1
 - A.1.1 DI21PART DBDA-1
 - A.1.2 PSB to Access DI21PARTA-2
 - A.1.3 FOCPSB to Access DI21PARTA-2
 - A.1.4 DI21PART Master FileA-2
 - A.1.5 DI21PART Access FileA-3
 - A.2 PATDB01 Sample.....A-3
 - A.2.1 PATDB01 DBDA-3
 - Database DBD for PATDB01A-4
 - Primary Index DBD for PATDB01A-4
 - Secondary Index DBDs for PATDB01A-5
 - A.2.2 PSB to Access PATDB01A-6
 - A.2.3 FOCPSB to Access PATDB01.....A-6
 - A.2.4 Master File to Access PATDB01A-7
 - A.3 EMPDB SampleA-8
 - A.3.1 EMPDB DBDsA-8
 - EMPDB01 DBD.....A-8
 - EMPDB02 DBD.....A-9
 - EMPDB03 DBD.....A-10
 - A.3.2 PSB to Access the EMPDB Databases.....A-11
 - A.3.3 FOCPSB to Access the EMPDB Databases.....A-11
 - A.3.4 Master Files to Access the EMPDB DatabasesA-12
 - EMPDB01 Master FileA-12
 - EMPDB02 Master FileA-13
 - EMPDB03 Master FileA-14
 - EMPDBJ Master FileA-15

A.4	AIHDAM Sample	A-16
A.4.1	AIHDAM DBD	A-16
A.4.2	PSB to Access the AIHDAM Database.....	A-17
A.4.3	FOCPSB to Access the AIHDAM Database.....	A-17
A.4.4	Master File to Access the AIHDAM Database.....	A-17
B.	Tracing Interface Processing	B-1
B.1	DLITRACE.....	B-2
B.2	Interface Traces	B-3
B.2.1	Allocating Interface Traces	B-3
B.2.2	Disabling Interface Traces.....	B-4
B.2.3	Batch Trace Allocation.....	B-4
B.2.4	FSTRACE Example	B-5
B.2.5	FSTRACE4 Statistics in the DBCTL Environment	B-13
C.	Release Dependent Interface Features	C-1
C.1	The SET IMS Command.....	C-1
C.2	Interface Environmental Commands.....	C-2
C.2.1	Commands for Implementing the DBCTL Environment.....	C-3
C.2.2	The IMS SET ? Query Command	C-4
C.3	Describing a Secondary Index Without Auto Index Selection.....	C-5
C.4	Fixed Format FOCPSBs.....	C-10
C.5	Accessing the BMP Extension	C-10
C.5.1	Example: Initiating a BMP Extension in DLI Mode	C-12
C.5.2	Invoking the Interface in the BMP Extension Environment	C-13
C.5.3	Terminating a BMP Extension	C-14
C.6	Accessing IMS Databases From CMS	C-15
D.	Installation Instructions	D-1
D.1	Pre-Installation and Maintenance Requirements.....	D-1
D.1.1	Software Requirements	D-2
D.1.2	Maintenance	D-2
D.2	Basic Installation Steps	D-3
D.2.1	Allocate the Interface Libraries	D-3
D.2.2	Unload the Distribution Tape	D-4
D.2.3	Prepare the Interface Run-time Libraries	D-5
	The MASTER Library.....	D-5
	The FOCPSB Library	D-6
	The ACCESS Library	D-6
	The FOCEXEC Library.....	D-7
	The Interface Load Library	D-7
	The ERRORS Library	D-8
	The Maintenance Library	D-8

Contents

- D.3 DBCTL InstructionsD-8
 - D.3.1 Test Your DBCTL Installation: IMDTSTD-9
 - D.3.2 Modify APPLCTN MacrosD-10
 - D.3.3 Create the DRA Startup Table: DFSPZPxxD-10
 - Choose the Suffix for the DRA Startup Table.....D-10
 - Assemble and Link the DRA Startup TableD-11
 - D.3.4 Create an MSO Configuration File and JCL for the DBCTL Environment.....D-14
 - Sample Configuration FileD-14
 - Sample JCLD-15
 - Console Commands for Controlling the DBCTL EnvironmentD-16
 - Shutting Down IMS in the DBCTL Environment.....D-17
- D.4 XMI Server InstructionsD-17
 - D.4.1 Allocate Communication FilesD-19
 - D.4.2 Create JCL for XMI Server JobsD-20
 - D.4.3 Shutting Down IMS in the XMI Server Environment.....D-21
- D.5 Run-time Requirements.....D-22
- E. Interface Errors and Messages E-1
 - E.1 Interface Messages E-1
 - E.2 Common User Errors E-7
- Glossary.....G-1
- Index..... I-1

Information Builders[®], the Information Builders logo, FOCUS[®], TableTalk[®], EDA/SQL[®], and FOCCALC[®] are registered trademarks of Information Builders, Inc.

IBM[®], IMS/ESA[®], VTAM[®], VM/ESA[®], DB2[®], and MVS[®] are registered trademarks of International Business Machines Corporation.

RACF[™], DRDA[™], Distributed Relational Database Architecture[™], MVS/ESA[™], SQL/DS[™], and CICS[™] are trademarks of International Business Machines Corporation.

CA-IDMS[®]/DB, CA-DATACOM/DB[®], CA-ACF2[®], and CA-TOP SECRET[®] are registered trademarks of Computer Associates, Inc.

Teradata[®] is a registered trademark of Teradata Corporation.

Oracle[®] is a registered trademark of the Oracle Corporation.

ADABAS[®] is a registered trademark of Software A.G.

SUPRA[®] and TOTAL[®] are registered trademarks of Cincom Systems, Inc.

MODEL 204[®] is a registered trademark of Praxis International, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 1997, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Preface

This manual describes how to use and install the FOCUS IMS/DB Interface. It is applicable to FOCUS Release 7.0, and it works with all IMS versions, subject to the constraint that certain features are available only in IMS version 3.1 and above; these cases are noted in the text.

With the FOCUS IMS/DB Interface installed, you can use FOCUS to analyze and report from data stored in IMS databases. The Interface provides read-only access to the IMS database management system (DBMS) under the MVS operating system; access from VM is available through the FOCUS Cross-Machine Interface.

If you also have the FOCUS DB2 Interface installed, you can access IMS databases from a DB2 MODIFY procedure. See the *FOCUS for IBM Mainframe DB2 and SQL/DS Read/Write Interface Users Manual* for details.

This updated manual includes corrections and FOCUS Release 7.0 features. Use it in conjunction with the *FOCUS for IBM Mainframe Users Manual*.

To use this manual effectively, you must be familiar with basic FOCUS reporting syntax and IMS concepts. You must also have access to IMS database information. This includes both data within the databases and descriptive information about them (such as database descriptions and field names). Check with your IMS database administrator (DBA) about file information, storage, and other site-specific considerations.

There is a fold-out glossary of standard terminology used throughout this manual on the inside back cover.

How This Document Is Organized

- Chapter 1, *Introduction*, describes how the Interface functions and how to use it.
- Chapter 2, *IMS Overview and Mapping Concepts*, provides an overview of IMS concepts and procedures for mapping IMS structures to FOCUS.
- Chapter 3, *Creating FOCUS Descriptions*, describes requirements for FOCUS Master Files, FOCPSBs, and Access Files.
- Chapter 4, *Reporting Efficiencies*, discusses advanced reporting topics such as optimization of screening conditions and the JOIN command.
- Chapter 5, *Environments*, describes the environments available with the Interface and provides sample CLISTs and JCL for invoking the Interface.
- Chapter 6, *Security*, describes IMS security.
- Appendix A, *Sample File Descriptions*, contains sample Master Files, FOCPSBs, DBDs, and PSBs used in the examples included throughout this manual.
- Appendix B, *Tracing Interface Processing*, describes IMS and Interface trace facilities.
- Appendix C, *Release Dependent Interface Features*, explains Interface environmental commands; it also describes techniques that were different in prior FOCUS Releases and are still supported.
- Appendix D, *Installation Instructions*, contains installation instructions.
- Appendix E, *Interface Errors and Messages*, lists Interface messages and includes a discussion of common user errors.
- The inside back cover contains a fold-out glossary of standard terminology.

Note: If you need additional information on how to use FOCUS, consult the *FOCUS for IBM Mainframe Users Manual*.

Documentation Conventions

The following conventions are used to describe command syntax in this manual:

UPPERCASE	Commands and required keywords are presented in uppercase and must be typed as shown. (In some cases, a shorter unique truncation is acceptable.)
lowercase	User-supplied parameters are presented in lowercase.
Punctuation	Required as shown.
—	Underscore indicates a default option.
{ }	Braces enclose groups of required parameters; select one.
[]	Brackets enclose optional parameters; none is required.
...	Horizontal ellipses indicate a continuation of syntax.
. . .	Vertical ellipses indicate intervening commands for syntax.

Note:

- At the command level, FOCUS accepts syntax in mixed case, uppercase, or lowercase but transmits it in uppercase.
- In sample sessions, FOCUS, Interface, IMS DBMS, or system responses are presented in uppercase; user responses are presented in lowercase.

Index Conventions

Please note the following conventions used throughout the index:

- Special character entries are listed first, followed by the remaining index entries in alphabetical order.
- These standard abbreviations apply:
 - MSO for Multi-Session Option.
 - PTF for program temporary fix.
- Commands appear in uppercase. For example, FIN instead of fin command.

New Features/Enhancements

The following chart summarizes the new features by chapter:

Chapter	New Features
2. <i>IMS Overview and Mapping Concepts</i>	<ul style="list-style-type: none">• Support for logical segment types (7.0.8).
3. <i>Creating FOCUS Descriptions</i>	<ul style="list-style-type: none">• Support for dynamic PCB selection (7.0.8).
5. <i>Environments</i>	<ul style="list-style-type: none">• Support for CICS Version 3.3 or higher (7.0.8).• Support for Non-stop Processing (7.0.8).
6. <i>Security</i>	<ul style="list-style-type: none">• Support for security exit IMSECHK (7.0.8).
B. <i>Tracing Interface Processing</i>	<ul style="list-style-type: none">• Support for FSTRACE2 (7.0.8).

Related Publications

Related publications include:

- *FOCUS for IBM Mainframe Users Manual Release 7.0 (DN1000983.0495).*
- *FOCUS for IBM Mainframe Multi-Session Option Installation and Technical Reference Guide Release 7.0 (DN1000966.1095).*
- *FOCUS for IBM Mainframe MVS/TSO Installation Guide Release 7.0 (DN1000994.0897).*
- *FOCUS COBOL FD Translator Users Manual Release 2.0 (DN1000023.0194).*
- *FOCUS for IBM Mainframe Cross-Machine Interface Users Manual Release 6.5 (DN0500005.0691).*
- *FOCUS for IBM Mainframe DB2 and SQL/DS Read/Write Interface Users Manual Release 7.0 (DN1000048.1195).*
- *FOCUS Summary of New Features CD, Release 7.0 (DN1001037.0597).*

Note: The title and Document Number (DN) information provided here are accurate as of this printing. To ensure up-to-date information when ordering, please consult the latest *Information Builders Publications Catalog*.

User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. Thank you, in advance, for your comments.

Customer Support

Questions about FOCUS, FOCUS products, or EDA/SQL?

Call Information Builders Customer Support Service (CSS) at (800)736-6130. Customer service representatives are available between 8:00 a.m. and 8:00 p.m. EST to address all your FOCUS and EDA/SQL questions.

To help our consultants answer your queries most effectively, please be ready to provide the following information when you call:

- Your six digit site code number (xxxx-xx).
- The FOCEXEC procedure (preferably with line numbers).
- Master File with picture (provided by CHECK FILE).
- Run sheet (beginning at login, including call to FOCUS), containing the following information:
 - ? RELEASE
 - ? FDT
 - ? LET
 - ? LOAD
 - ? COMBINE
 - ? JOIN
 - ? DEFINE
 - ? STAT
 - ? SET/? SET GRAPH
 - ? USE
 - ? TSO DDNAME OR CMS FILEDEF

- The exact nature of the problem: for example, are the results or is the format incorrect; does an abend occur; are the text or calculations missing or misplaced; is this related to any other problem?
- Has the procedure ever worked in its present form? Has it been changed recently?
- What release of the operating system are you using? Has it, FOCUS, or an interface system changed?
- Is this problem repeatable?
- Information Builders consultants can also give you general guidance on FOCUS capabilities and documentation.
- You can also FAX your questions to CSS at (212)564-1881, or upload them to the FOCWIZARD or FOCSERVICES forums on CompuServe.

1 Introduction

With the FOCUS IMS/DB Interface, you can use FOCUS to access IMS or DL/I databases. (From this point forward, IMS refers to IMS and/or DL/I.) FOCUS is well adapted to the IMS environment and fully supports its data model.

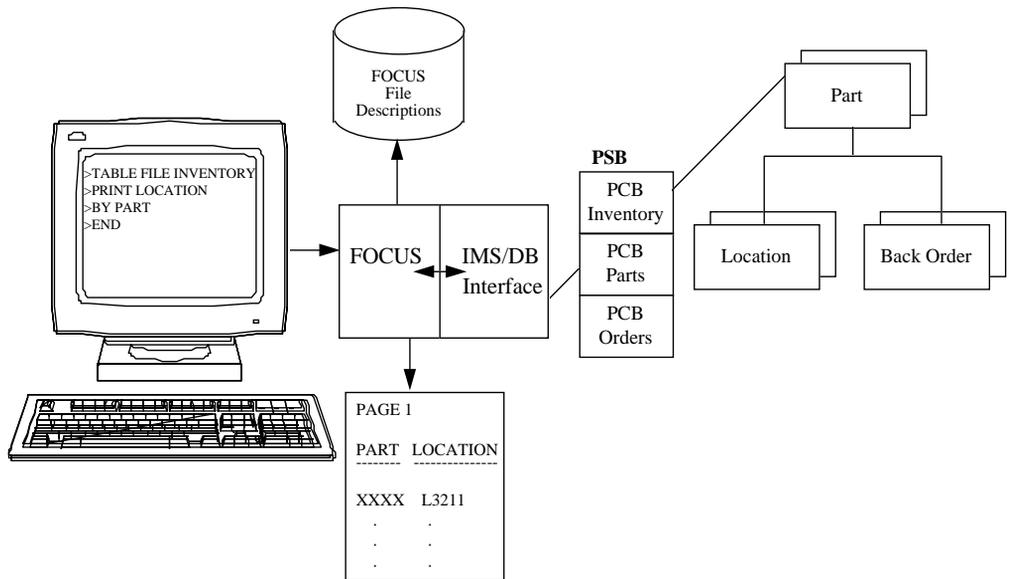
Beginners as well as advanced data processing professionals can take advantage of Interface data retrieval and analysis facilities comprehensive enough to satisfy virtually any requirement. FOCUS file descriptions integrate all facilities and provide transparent access to the underlying data file.

Since IMS and FOCUS both view databases as hierarchical multipath trees, FOCUS concepts map easily to those of IMS. For example, FOCUS represents IMS segments as FOCUS segments and IMS fields as FOCUS fields. You can represent an IMS repeating field as an OCCURS segment. To implement relationships between IMS files dynamically, you can use the JOIN command.

The Interface provides read-only access to existing IMS databases; as a result, FOCUS features that perform write operations, such as MODIFY and FSCAN, are not supported. The Interface uses only standard IMS read-only calls; it never jeopardizes the integrity of an IMS database. The Interface also supports certain IMS security features and complements existing IMS security. FOCUS DBA security features permit controlled data access at the user, file, field, or field-value levels.

When you issue a FOCUS retrieval request, the Interface translates it into an equivalent set of IMS DL/I (Data Language/I) calls. When the IMS DBMS returns data in response to the request, the Interface passes this data to the FOCUS Report Writer for formatting and, possibly, additional processing. The Report Writer can process data from any FOCUS-readable file. The Interface does not recreate the data in the form of a FOCUS database.

The following diagram depicts report processing:



FOCUS and the IMS DBMS interact as follows:

1. Given a report request, the Interface builds DL/I calls that define the request in terms the IMS DBMS can understand.
2. Having received these calls from the Interface, the IMS DBMS retrieves data targeted by the request and returns it to FOCUS.
3. The IMS DBMS sends records one at a time and/or a return code back to the Interface, which in turn passes it to FOCUS for further processing.

The Interface creates SSAs (Segment Search Arguments) to optimize DL/I calls based on request criteria and thus performs two primary tasks :

- Issuing retrieval instructions to obtain data from IMS databases.
- Establishing retrieval procedures that make efficient use of the available retrieval techniques.

Just as important, the Interface initiates and monitors communication between itself and the IMS DBMS and provides descriptive error messages when necessary.

1.1 Environments

The Interface operates in conjunction with FOCUS to access the IMS DBMS under two operating systems:

- The **MVS** operating system, specifically the TSO, MVS batch, and FOCUS Multi-Session Option (MSO) environments.
- The **VM** operating system and the CMS environment through the FOCUS Cross Machine Interface (XMI) for DL/I. The Cross-Machine Interface provides a means of running FOCUS under CMS while accessing IMS databases that exist under DOS/VSE or MVS on the same or separate physical machines. Channel-to-channel adapters, either virtual or real (dedicated), are used for the connection.

IMS versions 3.1 and above offer the DBCTL (Database Control) facility for connecting to IMS under MVS. The FOCUS IMS/DB Interface supports this facility and takes full advantage of its efficiency, security, and application control enhancements. Chapter 5, *Environments*, contains a complete description of each environment available for accessing IMS databases with the Interface.

The Interface is compatible with all versions of IMS; however, access through DBCTL is available only with IMS versions 3.1 and above.

If you have the appropriate FOCUS Interfaces installed, FOCUS can implement joins between IMS databases and other file types such as FOCUS databases, QSAM and VSAM files, DB2 tables, MODEL 204 databases, and CA-IDMS/DB databases.

1.2 Supported IMS Access Methods

IMS databases can be designed to use a variety of specialized access methods. The IMS access methods supported by the Interface include the following:

- HSAM
- HISAM
- HDAM
- HIDAM
- DEDB
- MSDB

Chapter 2, *IMS Overview and Mapping Concepts*, describes these access methods.

1.3 Ease of Use

With the Interface installed, you use the FOCUS language to request access to IMS databases. There is no need for specialized subroutines or embedded commands.

In IMS, each view of a physical or logical database is represented by a PCB (Program Communications Block). A PSB (Program Specification Block) is a collection of PCBs; it controls an application's access to databases and views. FOCUS, like any other application, must access IMS data through PCBs in a PSB:

- To make a view intelligible to FOCUS, you create a Master File that describes it in FOCUS terminology. This Master File makes it possible to refer to the individual fields of the IMS database via the FOCUS fieldname or the IMS fieldname.
- To inform FOCUS of which PCBs are included in a particular PSB, and to associate a Master File with each PCB in the PSB, create a FOCPSB.

In fact, once you have Master Files and FOCPSBs that describe IMS databases and PSBs, you can use all FOCUS reporting facilities such as the Report Writer, graphics, statistics, and FOCCALC to access the data. You can also use the FOCUS Dialogue Manager facility to create prompt-driven procedures for reporting from IMS databases. There is no need to use conventional computer languages like COBOL.

Database administrators (DBAs) or application developers often create Master Files and FOCPSBs; therefore, they may already be available at your site.

If you need to create a Master File and your site has a COBOL copy book that describes the IMS data, you may be able to use the COBOL FD Translator to help create the Master File. For more information, consult the *FOCUS COBOL FD Translator Users Manual*.

1.4 Efficiency

When you issue a FOCUS report request, the Interface analyzes the requirements specified in the request and sets up retrieval procedures to efficiently locate and retrieve the data records that fulfill those requirements. Depending on the specific request, the Interface automatically generates optimized DL/I calls.

The Interface retrieves from the IMS DBMS only those records corresponding to the fields referenced in the report request. Additionally, the Interface may instruct the IMS DBMS to apply the record selection criteria specified in the request, freeing FOCUS from this task. This optimization technique reduces the volume of DBMS-to-FOCUS communication, resulting in faster response times for Interface users. FOCUS can then join, sort, and aggregate data as necessary.

1.5 Security

All operating system security features or restrictions that are in effect apply to the Interface.

FOCUS respects all existing IMS security. That is, a user must be defined to the IMS DBMS as authorized to use the PSB that controls access to the data. This authorization must come from the IMS database administrator (DBA) via the application group name (AGN).

When using the DBCTL facility, the Interface supports access to standard security systems through the standard SAF interface. The SAF interface is supported by security products such as RACF, CA-TOP SECRET, and CA-ACF2. Before allowing access to a particular PSB, the system verifies that the user is authorized to read the PSB. Chapter 6, *Security*, discusses DBCTL security.

FOCUS also provides its own security facilities; you can use them as a complement to IMS security. For example, you can encrypt Master Files that contain security information. FOCUS security can enforce the following levels of restriction:

- File-level security to prevent access to an IMS database.
- Field-level security to limit the fields within an IMS database that are accessible to a user.
- Field-value security to limit the segments within an IMS database that are accessible to a user based on a specified field's values.

Thus, FOCUS provides the Database Administrator with a more sensitive security mechanism than does IMS, making it possible to use a few broadly sensitive Program Specification Blocks (PSBs) for a large class of users, rather than generating PSBs for every combination of access rights.

Refer to the *FOCUS for IBM Mainframe Users Manual* for information about DBA security.

2 IMS Overview and Mapping Concepts

This chapter discusses IMS and FOCUS concepts and explains how IMS elements correspond to their FOCUS counterparts. The concepts in this chapter apply to Master Files, FOCPSBs, and joins.

- Section 2.1, *Overview of IMS Concepts*, discusses IMS terms and concepts that are necessary for understanding how to describe IMS structures to FOCUS.
- Section 2.2, *Mapping IMS Elements to FOCUS*, discusses mapping concepts for individual elements.
- Section 2.3, *Mapping IMS and FOCUS Relationships*, discusses mapping concepts and procedures that explain how relationships are implemented.

2.1 Overview of IMS Concepts

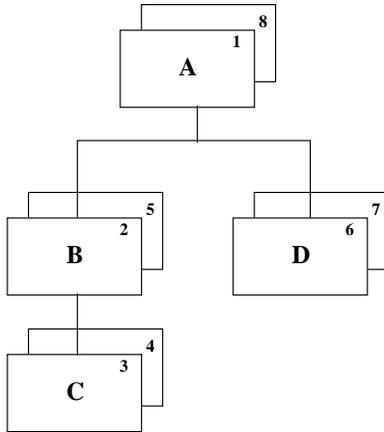
FOCUS requires a Master File that describes any database it accesses. Since IMS and FOCUS both view a database as a hierarchical multipath tree, describing a particular view of an IMS database to FOCUS is straightforward. However, IMS uses several specialized control blocks for regulating an application's access to its databases, and FOCUS requires additional information for negotiating its way through these control blocks.

The rest of this section explains the concepts necessary for retrieving information from IMS databases. Sections 2.2, *Mapping IMS Elements to FOCUS*, and 2.3, *Mapping IMS and FOCUS Relationships*, describe rules for implementing these IMS concepts in FOCUS. Chapter 3, *Creating FOCUS Descriptions*, provides detailed syntax rules for FOCUS file descriptions.

2.1.1 Hierarchical Structure

A hierarchical database is a collection of segments associated through parent-child relationships. Each segment is a child (or dependent) of the segment directly above it in the hierarchy and is the parent of all segments directly below it. A segment can have multiple children or no children, but it can have only one physical parent. Segment instances of the same type with the same parent are called *twins*. The root segment is the segment at the top of the tree; it has no parent.

The following diagram depicts a hierarchical database structure:



Each box represents a segment instance in the hierarchy. (Although the diagram depicts two instances of each segment type, there may be more or fewer instances of any type.) The numbers in the boxes indicate the order in which IMS accesses the segments when an application requests sequential access (for situations that allow sequential access). This ordering of segments is called the *hierarchical sequence*. A database record consists of a root segment instance along with all of its descendant segment instances in hierarchical sequence. The *hierarchical path* to a segment instance consists of the segment itself and all of its ancestors starting from the root.

A segment consists of fields. A field is the smallest logical unit of data that an application can request.

Sequence Fields

In IMS, a segment's key field is called the *sequence field*; it identifies the segment. A key is called unique if no two segments can contain the same key value.

The value of a segment's sequence field determines its position in its chain of twins (segments of the same type under the same parent). The primary key for an IMS database is the sequence field of its root segment. This key identifies and orders the records of the database.

In some IMS databases, the records are physically stored in root key order. In others, the roots are stored randomly and retrieved via a hash code (randomizing function) or an index.

Dependent segments are always retrieved by searching sequentially from the parent to the first instance of the child segment type and then through each occurrence of the child segment type, in order, until the required instance is found. In some types of IMS databases, dependent segments are physically stored in order of their sequence field (if there is one). In others, the sequence is maintained by pointers from one child to the next; however, even in this case, the instances are initially loaded in sequence field order.

Secondary Indexes

If you want to access a segment in order of a field that is not its sequence field, IMS provides the option of creating a secondary index on the field. In some situations, using a secondary index improves performance. The secondary index is itself a separate database; each of its records contains a value of the field to be indexed and a pointer to the target segment of the database record containing that value.

When using a secondary index, IMS locates a record by first reading the “index” database to retrieve the appropriate pointer and then using the pointer to read the “data” database.

Logical Relations

Another way to alter the order in which IMS views and retrieves segments is through logical relations. A logical relation associates segments from one or more databases as logical (virtual) parents and children. The Interface cannot distinguish a logical database from a physical database and can access both equally.

Search Fields

IMS database descriptions (DBDs) do not necessarily describe every field in the database. They include entries for all sequence fields, but other fields are optional.

When a DL/I retrieval request needs to locate specific database records, it can include selection criteria on particular fields. In some cases these *search fields* are not sequence fields. In order to reference them in a DL/I call, the DBD must include descriptions of these fields. See *Describing a Database: The DBD*, in Section 2.1.3, *Overview of IMS Control Blocks*, for more information about IMS database descriptions.

Symbolic Pointers

When IMS traverses a database, it may follow pointers from one segment to the next. In many cases the pointers are symbolic; that is, they are key field values rather than actual addresses. The *symbolic pointer* to a segment instance (also called its *concatenated key*) is the concatenation of the key values of all segments along the hierarchical path to that segment, starting from the root. All access to dependent segments is through the root.

2.1.2 Overview of IMS Access Methods

IMS supports several specialized access methods for storing and retrieving segments of a database. These can be subdivided into sequential and direct access methods. In the sequential methods, IMS maintains the hierarchical sequence by physically placing the segments in sequence. In the direct methods, IMS maintains the sequence with pointers. The Interface supports any IMS access method that allows command codes and qualified SSAs. The most common ones are:

- **HSAM (Hierarchical Sequential Access Method).** The database is stored physically in hierarchical sequence as on a tape. Updates require rewriting the entire database. No direct access is possible.
- **HISAM (Hierarchical Indexed Sequential Access Method).** Initial loading of the database is in physical sequence with the tail ends of records that are too long going into an overflow area. All insertions after initial load are in the overflow area. There is an index for direct access to the root segment. The sequence field, or key, must be unique; that is, no two root segments can contain the same key value.
- **HDAM (Hierarchical Direct Access Method).** Root segments are inserted by applying a randomizing routine to the sequence field; the value computed assigns a storage location to the segment. Root segments are retrieved by applying the same hash code. Descendant segments are stored independently and linked by pointers. Since the roots are stored in a random physical sequence without an index, they cannot be accessed sequentially in root key order.
- **HIDAM (Hierarchical Indexed Direct Access Method).** The database consists of two parts: the “data” database and a separate database that is an index on the sequence field of the root segment. The “data” database is physically loaded in hierarchical sequence. Roots can be accessed sequentially or directly, but each direct access requires reading the “index” database prior to reading the “data” database. Root sequence fields must be unique; that is, no two root segment instances can contain the same key value.
- **DEDB (Data Entry Database).** This is a Fast Path access method. *Fast Path* is an option that provides enhanced data reliability and availability and improved response time in exchange for limitations on the structure of a database and on its ability to take advantage of techniques like secondary indexing and logical relationships. DEDBs can have special segments called sequential dependents that segregate high volume data and make loading the database more efficient. DEDBs can also be subdivided into areas, each of which contains all segment types for particular roots, thus partitioning the database by root key values. (This differs from dataset groups that are available for many types of databases and that segregate particular types of segments for *all* roots.) The advantages include the ability to create very high volume databases, to make most of the database available even if an area is undergoing maintenance, and to replicate areas for increased availability and problem recovery.

- **MSDB (Main Storage Database).** This is a Fast Path access method that limits the database structure to root-only databases with no logical relationships, secondary indexes, variable length data, or field level sensitivity. The major advantage of MSDBs is the extremely fast access that IMS achieves by loading and accessing them in main storage in the online (IMS/DC) region. You can access MSDBs only through the BMP mode of program DFSRRC00 (see Chapter 5, *Environments*), because they exist in the IMS/DC region.

2.1.3 Overview of IMS Control Blocks

IMS uses special control blocks for describing a database, regulating an application's access to databases, and communicating with an application. This section briefly describes these control blocks.

Describing a Database: The DBD

The DBD (Data Base Description) is a control block that describes the structure of a physical or logical database. It contains information necessary for locating particular segments, specifies access method and ddname allocation information, and describes the hierarchical structure of the database.

To describe the structure of the database, the DBD contains SEGM and FIELD statements:

- SEGM statements name the segments and their parents and specify their lengths and pointer types.
- FIELD statements name the fields, specify their positions within the segment, describe their data types, identify whether they are sequence fields, and, if they are sequence fields, specify whether they are unique or non-unique. FIELD statements are not required for all fields in the database. They are required for sequence and search fields.

If the database participates in a logical relation, the DBD may include logical child information.

After the actual database description, the DBD includes a DBDGEN statement that instructs IMS to take the user-provided DBD source statements and create a load module that the system can use. Appendix A, *Sample File Descriptions*, illustrates sample DBDs.

Defining an Application's Access to Databases: The PSB

The PSB (Program Specification Block) contains information about an application's authorization to use databases. Each view of a database that the application can access is described within the PSB; this description is called a PCB (Program Communication Block, described in the following section). Therefore, the PSB is simply a collection of PCBs.

The PSB can contain two types of PCB; one type provides access to databases (TYPE=DB) and the other type implements teleprocessing communication and batch checkpointing (this type is called an I/O PCB, TYPE=TP). The PSB can contain duplicate PCBs for maintaining multiple positions within a database or for performing recursive joins. The PSB can also include multiple PCBs for the same database in order to provide different views of the database or to allow different types of access to the database.

The PSB ends with a PSBGEN macro statement that contains information about the PSB, such as its name; the PSBGEN creates a load module from the source statements. Appendix A, *Sample File Descriptions*, illustrates sample PSBs.

Describing a Database View and Communicating With the DBMS: The PCB

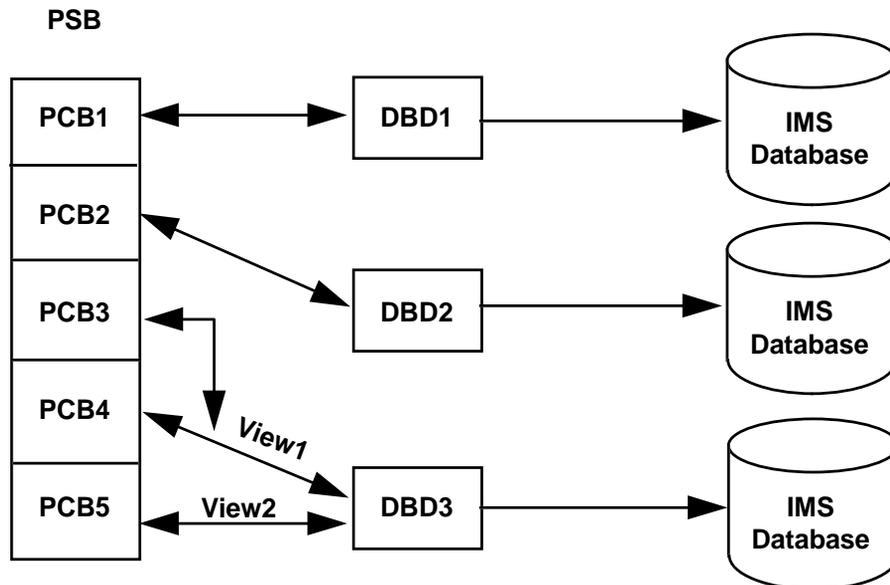
A PCB (Program Communication Block) has several functions:

- It describes a view of a database; that is, it names the database to be accessed (the DBDNAME parameter), lists the segments that a program can access through that PCB (the SENSEG statement), and names the parent of each sensitive segment (the PARENT parameter).
- It describes the type of access to the database that a program can have through that PCB (the PROCOPT parameter). For example, it may limit requests to retrieval only.
- It may specify that a secondary index is to be used as the main entry point into the database (the PROCSEQ parameter).
- It keeps track of position within the database; that is, it remembers which segment was retrieved the last time the PCB was used.
- It receives a status code from the DBMS about the results of each call it makes to the database.

In a PSB, the PCBs are listed one after another prior to the PSBGEN statement. I/O PCBs must come before database PCBs. A segment cannot be included in a PCB unless its parent is also included. For the root segment, PARENT=0.

An application program can use all PCBs in a PSB concurrently. The program gives a name to each PCB and defines its structure by applying a mask that allocates program variables to receive the status information returned by IMS. Although IMS returns status information to the PCB, it does not place the retrieved database segment into the PCB. The segment goes into an I/O area identified in the DL/I call (see Section 2.1.4, *Overview of DL/I Calls*). For PCB examples, refer to the PSB samples in Appendix A, *Sample File Descriptions*.

The following diagram illustrates the relationship between DBDs, PSBs, and PCBs.



Key Sensitivity

All access to segments within an IMS database proceeds from the root segment through the hierarchical path to the desired segment. When an application has no use for the data in a segment, but does need data from one of its children, the PCB can specify PROCOPT=K to make the parent segment *key sensitive*. This gives IMS access to the key value of the segment but instructs it not to return any data from the segment to the application.

Field Level Sensitivity

A PCB can include an optional list of sensitive fields for a segment (the SENFLD statement). If it does, IMS returns only those fields to the application, not the entire segment. This *field level sensitivity* provides data independence and security. Even if the segment changes, the PCB and application program can remain the same; also, for security purposes, application programs can only access specified fields.

Status Codes

IMS stores a status code in the PCB after each DL/I call. The Interface checks the status code after each call to determine whether the call was successful and if not, why not.

Key Feedback Area

After a successful call, the *key feedback area* in the PCB contains the *concatenated key* of the retrieved segment (the keys of each segment in the hierarchical path to the retrieved segment).

The ACB

An ACB is an optimized PSB that contains a combination of information from the PSB and the DBD; however, even with an ACB, the normal PSB is still required. ACBs are created by an ACBGEN and are used to access online databases.

2.1.4 Overview of DL/I Calls

To access an IMS database, an application program must call a special DL/I subroutine, such as ASMTDLI (Assembler to DL/I). Each DL/I call passes IMS the following arguments:

- A function code that defines the type of call. The Interface makes only two types of calls: GU (Get Unique) and GN (Get Next).
- The PCB to use for the call.
- The I/O area in which to put the retrieved segment.
- Segment search arguments (SSAs) that describe the desired segment. The number of SSAs in a call depends on the level of the segment to be retrieved and the type of call. The following section discusses SSAs in more detail.

Segment Search Arguments

If you want to retrieve a specific segment or type of segment, you must tell IMS how to find it. You do this by means of segment search arguments (SSAs).

There are two types of SSAs:

- An **unqualified SSA** consists of a segment name. Any segment of that segment type satisfies the SSA.
- A **qualified SSA** is a Boolean expression that defines an acceptable value or range of values for one or more fields in the segment. The fields referred to in the SSA must be search fields; that is, they must be listed in FIELD statements in the DBD. Only segments containing the proper values satisfy a qualified SSA.

SSAs can also include command codes that alter the way in which IMS completes the call. For example, the FIRST command code (*F) instructs IMS to begin its search at the start of the twin chain for that segment type, under the current parent, even if the PCB is positioned past that point. The Interface uses only one command code, the *U (parentage) command code. Appendix B, *Tracing Interface Processing*, includes a trace example that demonstrates the use of this command code.

Get Calls

The Interface retrieves data from IMS using two types of Get calls:

- **GU (Get Unique)** always starts from the beginning of the database and finds the first segment that satisfies all of the SSAs. It uses the index or hash code to locate an appropriate root segment. The segment type named in the last SSA is the type of segment that IMS retrieves and places in the I/O area.
- **GN (Get Next)** provides sequential retrieval. It keeps track of which segment was last retrieved (the *current database position*) and goes on from that point. It can be issued with or without SSAs. If there are no SSAs, it retrieves the next segment listed in the PCB in hierarchical sequence; this segment can be any type of segment. If there are SSAs, it retrieves the next segment that satisfies all of the SSAs; the last SSA determines the type of segment that IMS retrieves and places in the I/O area. Since HDAM roots cannot be retrieved in root key order, using GN calls on HDAM roots retrieves them in the random physical order in which they were loaded.

If a segment instance satisfies an SSA, but the current database position is past that segment in hierarchical sequence, GN will not retrieve the segment.

For a discussion of how the Interface creates SSAs, see Chapter 4, *Reporting Efficiencies*.

2.2 Mapping IMS Elements to FOCUS

To access an IMS database from FOCUS, you must describe IMS entities in FOCUS terms. Chapter 3, *Creating FOCUS Descriptions*, explains specific syntax requirements, and Appendix A, *Sample File Descriptions*, includes sample descriptions. This section presents an overview of the mapping concepts.

Note: The diagrams in this chapter present relevant portions of DBDs, PSBs, PCBs, and Master Files. They are not meant to contain all syntax elements or keywords.

2.2.1 Describing the PSB: The FOCPSB

A FOCPSB contains attributes (keyword=value pairs) that identify the PCBs in a PSB and associate each PCB with the name of a Master File. Section 2.2.2, *Describing the Database: The Master File*, and Chapter 3, *Creating FOCUS Descriptions*, describe the Master File in detail.

You associate a FOCPSB with the PSB it describes by giving them both the same name. FOCPSBs are stored as members of a partitioned dataset. The member name for a FOCPSB must be identical to the name of the corresponding IMS PSB.

Note: FOCPSBs created in prior FOCUS Releases may consist of fixed format records with no attribute keywords; Appendix C, *Release Dependent Interface Features*, discusses fixed format FOCPSBs. While this earlier format is still supported, the comma-delimited format is preferable.

In the FOCPSB, you must provide the following for each PCB:

- **The PCBNAME.** This value is the name of the corresponding Master File. A blank indicates either an I/O PCB (see Section 2.1.3, *Overview of IMS Control Blocks*) or a PCB that you do not want to access. You can include the same PCB multiple times in the PSB; each such duplicate should use the same Master File name.
- **The PCBTYPE.** This value identifies the type of PCB; acceptable values are DB for database PCBs, TERM for I/O PCBs, and SKIP for PCBs you will not access.

Important: SKIP is a reserved word for the FOCPSB. *Never* use SKIP as a Master File name.

Chapter 3, *Creating FOCUS Descriptions*, discusses additional attributes used for partitioning and concatenating PCBs.

The following diagram illustrates the correspondence between an IMS PSB and an equivalent FOCPSB:

1) IMS PSB (member PSB1)	2) FOCPSB (member PSB1)
PCB TYPE=DB,DBNAME=DBD1	PCBNAME=FILE1 ,PCBTYPE=DB
PCB TYPE=DB,DBNAME=DBD2	PCBNAME= ,PCBTYPE=SKIP
PCB TYPE=DB,DBNAME=DBD3	PCBNAME=MYFILE ,PCBTYPE=DB
PSBGEN PSBNAME=PSB1	

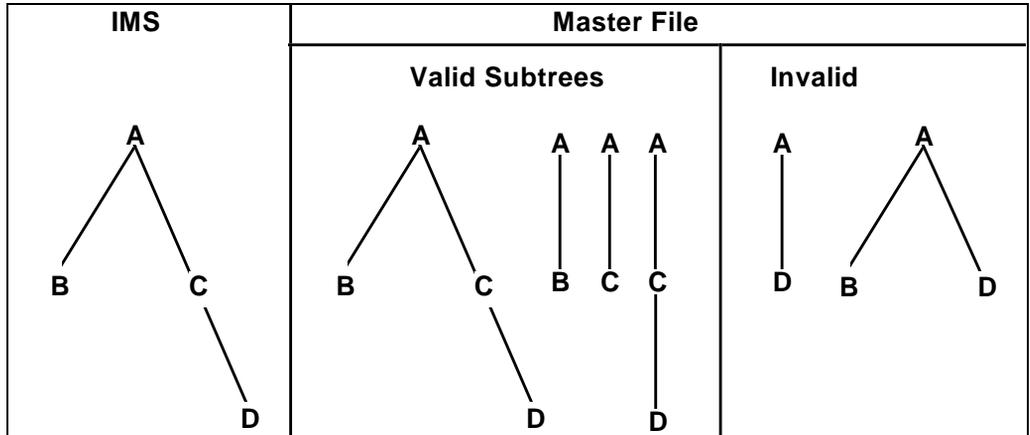
1. Is an IMS PSB named PSB1. It contains three database PCBs.
2. Is the corresponding FOCPSB, member PSB1 in the FOCPSB dataset. This FOCPSB ignores the second PCB in the PSB (PCBTYPE=SKIP); it can issue a report request through the first PCB with the syntax TABLE FILE FILE1 (against Master File FILE1), and through the third PCB with the syntax TABLE FILE MYFILE (against Master File MYFILE).

2.2.2 Describing the Database: The Master File

With the IMS/DB Interface, a Master File is not necessarily a complete description of the database, but rather is a description as seen through a specific PCB. If the PCB is not sensitive to a segment listed in the DBD, the Master File cannot include that segment. Therefore, in order to create a Master File for the PCB, you must combine information from the DBD and the PCB.

You do not have to describe every segment from the PCB in the Master File. However, the portion of the hierarchy you describe must be a subtree starting from the root. In a *subtree*, when you include a child segment, you must also include its parent.

The following diagram illustrates the concept of a subtree:



This section illustrates where each element that goes into the Master File comes from in the IMS schema. Chapter 3, *Creating FOCUS Descriptions*, discusses Master File syntax in detail.

In the following discussion, you will notice the following:

- An IMS field is equivalent to a field in the Master File.
- An IMS segment corresponds to a segment in the Master File.
- IMS fields that are composed of multiple elementary fields can be represented as GROUP fields in the Master File.
- IMS segments that have multiple definitions can be represented in the Master File with the RECTYPE attribute.
- IMS variable length segments and segments with repeating fields can be represented with an OCCURS segment in the Master File.

Note: Master Files should maintain the hierarchy of the structure as defined in the IMS DBD. The structure is traversed from top to bottom, left to right. Failure to maintain the hierarchy can produce unpredictable results.

Identifying the IMS Database

Each Master File corresponds to one PCB and each PCB accesses one DBD. The FILENAME value in the Master File can be any one- to eight-character name. However, for consistency and documentation purposes, the examples in this manual use the DBD name as the FILENAME value in the Master File:

<p data-bbox="321 1475 464 1501">1) IMS PSB</p> <p data-bbox="349 1562 721 1588">PCB TYPE=DB,DBNAME=DBD1</p>	<p data-bbox="785 1475 949 1501">2) Master File</p> <p data-bbox="813 1562 1021 1588">FILENAME=DBD1</p>
--	---

1. Is an IMS PSB containing a PCB for DBD1.
2. Is the FILE record of the Master File for that PCB. The FILENAME attribute has the value DBD1.

Describing IMS Segments to FOCUS

A Master File contains Segment records to describe the hierarchy of segments. These correspond to the SENSEG records in the PCB.

The Segment record in the Master File contains the following information:

- **The SEGNAME attribute.** Its value is the segment name from the SENSEG record in the corresponding PCB.
- **The SEGTYPE attribute.** Its value indicates whether the segment is a unique segment (there can be at most one segment instance for each parent), whether a segment is key sensitive, and whether the key (if there is one) is unique or non-unique. Recall that a key sensitive segment is used only for access to lower level segments and is specified by the parameter PROCOPT=K in the PCB. The information about a segment's key comes from the DBD. The FIELD record in the DBD specifies NAME=(name,SEQ,M) for a non-unique key field, and NAME=(name,SEQ,U) or (name,SEQ) for a unique key.

The following table lists permissible SEGTYPEs:

SEGTYPE	Definition	PROCOPT=K In PCB?	NAME= From DBD
U	Data sensitive, unique segment	No	Not Applicable
S0	Data sensitive, no key	No	(name)
S or S1	Data sensitive, non-unique key	No	(name,SEQ,M)
S2	Data sensitive, unique key	No	(name,SEQ,U) or (name,SEQ)
SH or SH1	Key sensitive, non-unique key	Yes	(name,SEQ,M)
SH2	Key sensitive, unique key	Yes	(name,SEQ,U) or (name,SEQ)

- **The PARENT attribute.** Its value is the name of the segment's parent from the SENSEG record in the PCB. The only difference is in the root segment. The PCB specifies PARENT=0 for the root segment or omits the PARENT parameter. In the Master File, you can specify the PARENT attribute of the root segment as PARENT= , or you can omit it.

The following diagram illustrates how to create a Segment record in the Master File:

1) IMS PSB	2) IMS DBD	3) Master File
PCB DBDNAME=DBD1	DBD NAME=DBD1	FILENAME=DBD1
SENSEG NAME=SEG1, PARENT=0,PROCOPT=GO	SEGM NAME=SEG1, PARENT=0	SEGNAME=SEG1,
	FIELD NAME=(FLD1,SEQ,U)	SEGTYPE=S2
SENSEG NAME=SEG2, PARENT=SEG1,PROCOPT=GO	SEGM NAME=SEG2, PARENT=SEG1	SEGNAME=SEG2, PARENT=SEG1,
	FIELD NAME=FLD2	SEGTYPE=S0

1. Is an IMS PSB that has a PCB with two sensitive segments, SEG1 and SEG2. SEG1 is the root segment (PARENT=0) and SEG2 is a child of SEG1.
2. Is the IMS DBD that the PCB is viewing. It indicates that the key for SEG1 is unique. There is no key specified for SEG2. FLD2 is a search field but not a sequence field.
3. Is the Master File corresponding to the PCB.

SEG1 is the root; therefore the PARENT attribute can be omitted. Since the segment is data sensitive and the sequence field is unique, SEGTYPE=S2.

SEG2 is a child of SEG1; therefore its PARENT=SEG1. Since it is not key sensitive and has no key field, SEGTYPE=S0.

Describing IMS Fields to FOCUS

To describe data fields in the Master File, you must also consider information from both the PCB and the DBD.

If the PCB you are describing contains SENFLD records for a segment, the Master File can view only the fields explicitly named in those SENFLD records.

However, if the PCB does not contain any SENFLD records for a segment, you can describe the entire segment in the Master File. You can get information about sequence and search fields from the DBD; to describe other fields you may have to refer to the COBOL FD description for the segment.

For each field you describe in the Master File you must include the following:

- **The FOCUS FIELDNAME.** This can be any name that complies with FOCUS field naming conventions, as described in Chapter 3, *Creating FOCUS Descriptions*.
- **The ALIAS.** The Interface uses the alias to distinguish between IMS sequence fields, IMS search fields, key fields from the root of an HDAM database, and other fields. These designations help the Interface produce optimized SSAs, as described in Chapter 4, *Reporting Efficiencies*.

If the field is not listed in the DBD, the alias can be any name that complies with FOCUS naming conventions.

If the field is listed in the DBD (it is a sequence or search field), the alias value takes the form IMSname.suffix. IMSname is the field name specified in the DBD. The table below lists suffix values:

Suffix	Description
KEY	IMS key field
IMS	IMS search field
HKY	Key of root of an HDAM database

Note: If you do not specify a key field (.KEY) for a keyed SEGTYPE (for example, S1), a FOC4277 message will result.

- **The USAGE format.** The USAGE format is the FOCUS display format for the field, as described in the *FOCUS for IBM Mainframe Users Manual*.
- **The ACTUAL format.** The ACTUAL format describes how the data is stored in the IMS database. The DBD specifies this information with the TYPE and BYTES parameters in each FIELD record. See Chapter 3, *Creating FOCUS Descriptions*, for a discussion of ACTUAL formats.

The DBD indicates the length of each segment and the length and starting position of each listed field within a segment. The Master File need not describe all fields from a segment, but it must include filler fields that occupy the same space as any field it omits. You can use the BYTES and START parameters from the DBD to determine how the fields in the segment are arranged. If you want to describe two fields that are separated by data that you do not need, you must include a field in the Master File occupying the unneeded space in order to avoid a gap.

IMS Overview and Mapping Concepts

The following diagram illustrates FIELDNAME and ALIAS values in a Master File for a HIDAM database:

1) IMS PSB	2) IMS DBD	3) Master File
PCB DBDNAME=DBD1	DBD NAME=DBD1	FILENAME=DBD1
SENSEG NAME=SEG1	SEGM NAME=SEG1	SEGNAME=SEG1,SEGTYPE=S2
SENFLD NAME=FLD1	FIELD NAME=(FLD1,SEQ,U) FIELD NAME=FLD2	FIELDNAME=MFDFLD1, ALIAS=FLD1.KEY
SENSEG NAME=SEG2	SEGM NAME=SEG2, BYTES=100	SEGNAME=SEG2,SEGTYPE=S2
	FIELD NAME=(FLD3,SEQ,U), BYTES=10, START=1	FIELDNAME=MFDFLD3, ALIAS=FLD3.KEY
	FIELD NAME=FLD4, BYTES=10, START=20	FIELDNAME=FILL1, ALIAS=
		FIELDNAME=MFDFLD4, ALIAS=FLD4.IMS
		FIELDNAME=OTHERINFO, ALIAS=OTHER

1. Is an IMS PSB. SEG1 has a SENFLD record for FLD1 limiting the view to that field alone. SEG2 has no SENFLD records; therefore, the entire segment is available.
2. Is the IMS DBD that the PCB is viewing. SEG1 has two fields listed, but the PCB is sensitive only to the first. SEG2 has two fields listed, but they do not describe the entire segment.
3. Is the corresponding Master File. In SEG1, only FLD1 is available because of the SENFLD record in the PCB. The FIELDNAME can be any convenient name. Since this field is the key for the segment, the alias value is the name from the IMS DBD with the suffix value KEY appended: ALIAS=FLD1.KEY.

In SEG2, the whole segment is available since the PCB has no SENFLD records for the segment. FLD3 is the key field; therefore its alias value is FLD3.KEY. FLD4 is a non-key search field; its alias value is FLD4.IMS. No other fields are described in the PCB, but the Master File defines two more fields from the segment. Their alias values are arbitrary names with no suffix values. Notice the filler field between FLD3 and FLD4.

IMS key fields, secondary index fields, and search fields can consist of multiple elementary fields. In the Master File, you can break the IMS field into component parts (field redefinition) using a GROUP field.

The following diagram illustrates a group key:

1) IMS DBD	2) Master File	3) User Request
<p>FIELD NAME=(G,SEQ,U), BYTES=8, TYPE=C</p>	<p>GROUP=G1, ALIAS=G.KEY, USAGE=A8, ACTUAL=A8</p> <p>FIELDNAME=F1, ALIAS=F1, USAGE=A4, ACTUAL=A4</p> <p>FIELDNAME=F2, ALIAS=F2, USAGE=A4, ACTUAL=A4</p>	<p>WHERE G1 EQ 'ABCD'/'WXYZ'</p> <p>WHERE F1 EQ 'CDEF'</p>

1. Is a DBD. The field G is 8 bytes long and is type character.
2. Is the Master File. It defines the group as an 8-byte alpha field. It then breaks this field down into two 4-byte alpha fields.
3. Is a FOCUS session or FOCEXEC that uses the group field and its high-order component in screening tests. A slash must separate components of the group (see Chapter 3, *Creating FOCUS Descriptions*).

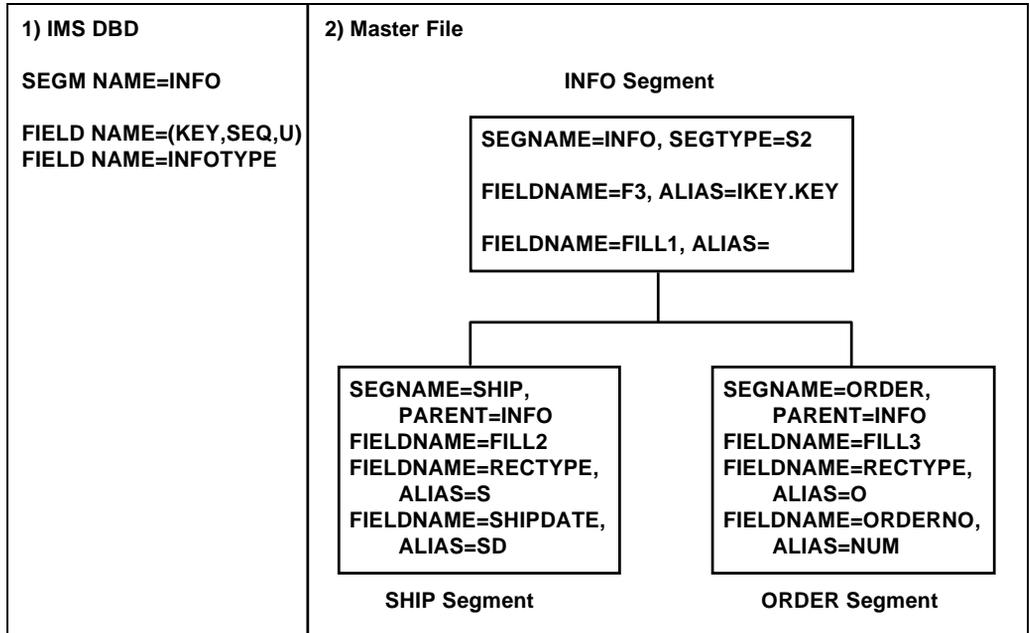
Describing IMS Segments That Have Multiple Definitions to FOCUS

An IMS segment can have multiple definitions. For example, a segment may contain either shipment or order information, depending on the value of one of its fields. If the field that identifies the type of segment is at the same position in each redefinition, you can use the RECTYPE attribute to define the different segment types in the Master File:

1. First define the non-changing portion of the segment as usual. Include a filler field for the part that will be redefined. The field that identifies the different types must be in the redefined portion.
2. Next, describe each redefined portion as a segment whose parent is the non-changing segment. Do not define a SEGTYPE for these children. Describe the field that identifies the segment type as FIELDNAME=RECTYPE in the Master File. The alias for the RECTYPE field is the value that identifies the type of data in the segment (in this example, either ALIAS=S, for a shipment, or ALIAS=O, for an order).

Include a filler field in each redefinition to occupy the fields that are in the non-changing segment.

The following diagram illustrates segment redefinition in the Master File:



1. Is a segment named INFO that has two definitions, one for shipment information and one for order information. Field INFOTYPE contains the value S in those segment instances that contain shipment data; it contains the value O in those segment instances that represent order information.
2. Is the portion of the Master File that represents the IMS segment. It uses three segments to describe the one IMS segment.

The parent segment, named INFO to match the IMS segment name, contains the key field since the key is not in the redefined portion. There is also a filler field to account for the redefined portion of the segment.

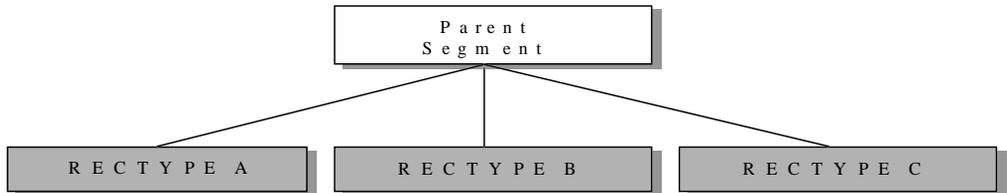
Each child segment has INFO as its parent and a null SEGTYPE value. In each child, the field that corresponds to INFOTYPE from the DBD has FIELDNAME=RECTYPE. The ALIAS value for the RECTYPE field in each child segment is the INFOTYPE value that identifies that type of segment: S for the SHIP segment type and O for the ORDER segment type. (Each segment type also has a filler to occupy the positions of the fields that are described in the parent segment, and each segment type describes additional fields that it needs for either order or shipment data.)

Chapter 3, *Creating FOCUS Descriptions*, explains how to describe a segment type that allows multiple RECTYPE values.

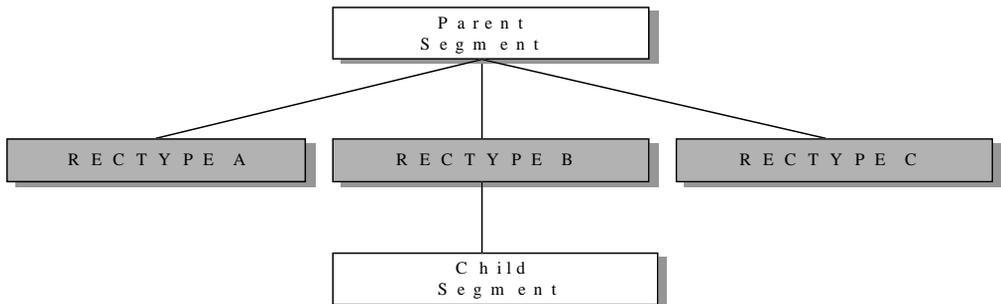
Supporting Logical Segment Types

In prior releases of the FOCUS Interface to IMS, logical segment types could not have dependent or child segments.

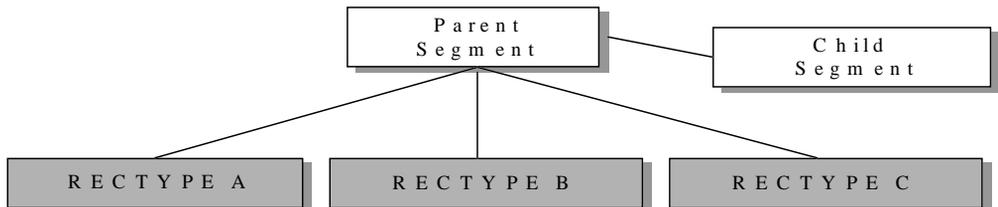
A segment defined as type RECTYPE is considered a logical segment type since it is possible that it will not exist for all records in a file. Previously, you had to define such a segment in the Master File at the lowest level of the hierarchy. For example:



As of the current release, the IMS Interface allows logical segment types to have child segments. In the following example, the RECTYPE B segment has a real segment described as a child. This child segment only exists when the RECTYPE field of the record specifies a B:



The previous example illustrates a child segment that only exists when there is an instance of RECTYPE B. The following example shows a child segment that exists for all record types. The child segment will be accompanied by data that depends on the record type:



Through the use of the record type, the Interface activates only the necessary paths. This eliminates unnecessary, independent paths from the hierarchy traversal. This efficient data access results in improved Interface performance.

Describing Variable Length Segments to FOCUS

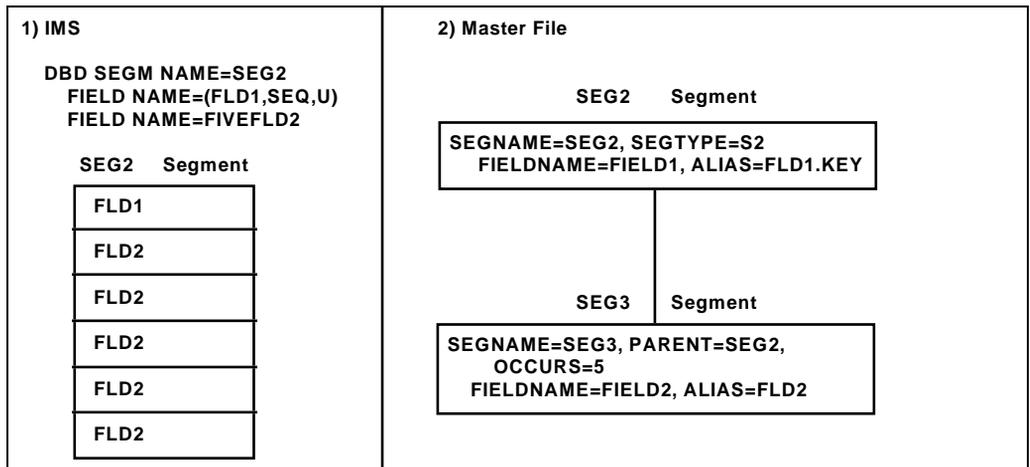
In an IMS database, segments can have repeating fields or repeating groups of fields. The number of repetitions may be fixed, may depend on the value of a field from the parent segment or from the non-repeating portion of the variable segment, or may have to be calculated from the segment length.

In the Master File, you describe a segment with repeating fields by defining each repeating field (or group) as a separate segment whose parent is the non-repeating portion of the segment. The child segment definition has no SEGTYPE, but it includes the OCCURS attribute to specify how many times the field repeats. The table below lists permissible values:

OCCURS=	Description
n	Is the number of times the field repeats in the segment.
fieldname	Is the name of a field whose value indicates the number of times the field repeats in the segment.
VARIABLE	Indicates that the number of repetitions must be computed from the length of the segment. In this case, the segment must contain a counter field as its first field; the counter field's alias in the Master File must be IMSname.CNT.

If the repeating field is not at the end of the segment, you must also identify its position within the segment (see Chapter 3, *Creating FOCUS Descriptions*).

The following diagram illustrates an OCCURS segment in the Master File. Chapter 3, *Creating FOCUS Descriptions*, includes examples for each value of the OCCURS attribute.



1. Is an IMS segment with one key field and one repeating field.
2. Are the equivalent segments in the Master File. Segment SEG2 contains the non-repeating portion of the IMS SEG2 segment. Segment SEG3 contains the repeating field; it specifies OCCURS=5. SEG3 has no SEGTYPE value.

Describing a Secondary Index to FOCUS

Using IMS secondary indexes, you can retrieve records in order of a field other than the key field. (A secondary index is itself a database with its own DBD.) The DBD for a database that uses a secondary index includes an XDFLD statement that assigns a field name to the index.

If a PCB includes the parameter PROCSEQ=indexDBDname, the named index is used as the main entry point into the database.

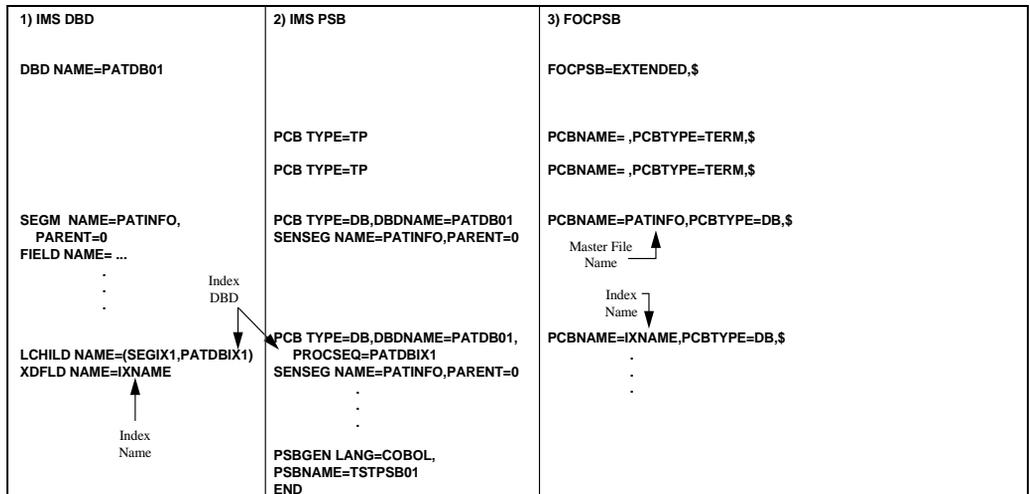
Starting with FOCUS Release 7.0, one Master File can describe the primary index and multiple secondary indexes. You must also include a record for each secondary index in the FOCPSB. Then, when you issue a FOCUS report request, the Interface inspects all key fields and secondary indexes to select the optimal retrieval path based on the selection criteria in the request.

In prior releases, each secondary index required a separate Master File, and the application programmer had to decide which Master File to use with each request. Appendix C, *Release Dependent Interface Features*, describes that technique.

In order to access a database through a secondary index, the IMS PSB must contain a PCB that defines the index as the main entry point into the database. The PCB does this by identifying the DBD for the index in the PROCSEQ parameter (recall that a secondary index is, itself, an IMS database). Of course, the Interface requires the IMS PSB also to include a PCB for the normal entry point into the database; this PCB does not include a PROCSEQ parameter.

The FOCPSB has a one-to-one correspondence with the PSB. The FOCPSB entry that corresponds to the normal entry point into the database supplies the name of the Master File. Each FOCPSB entry that corresponds to a secondary index PCB supplies the name of the index from the XDFLD record of the DBD.

The following diagram illustrates an IMS PSB and its corresponding FOCPSB:

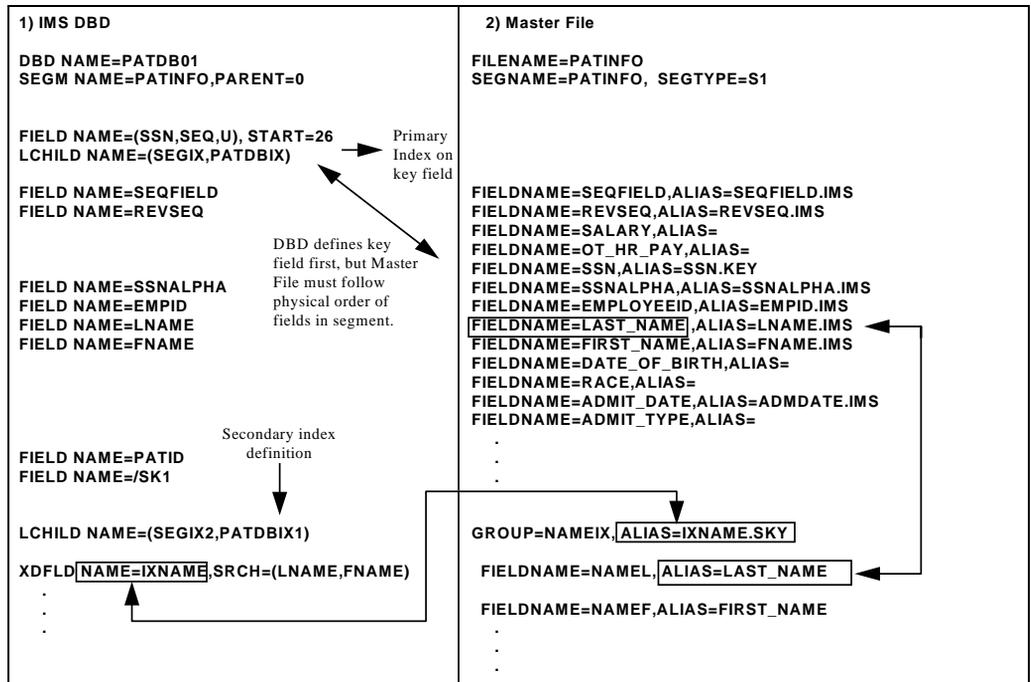


1. Is the IMS DBD for the PATDB01 database. The LCHILD record for the secondary index gives the name of the index DBD. The corresponding XDFLD record names the index.
2. Is an IMS PSB with two PCBs for PATDB01. The first PCB does not include a PROCSEQ parameter; therefore, it uses the normal entry point into the database. The second PCB includes the parameter PROCSEQ=PATDBIX1. Therefore, a secondary index is the main entry point into the database through this PCB, and the PATDBIX1 DBD describes it.
3. Is the FOCPSB that describes the IMS PSB. The entry corresponding to the first PCB supplies the Master File name, PCBNAME=PATINFO. The entry corresponding to the PCB for the secondary index provides the index name, PCBNAME=IXNAME.

To define the secondary index in the Master File, you must do the following:

- Describe each field that participates in a secondary index as a key or search field, with the suffix .KEY, .HKY, or .IMS.
- Describe the rest of the IMS segment with FOCUS field definitions or filler fields.
- At the *end* of the root segment, define each secondary index as a group field:
 - The ALIAS of the group field must have the suffix .SKY (Secondary index key) appended to the index name from the XDFLD record in the DBD; that is, ALIAS=XDFLDname.SKY, where XDFLDname is the value of the NAME parameter in the XDFLD record of the DBD.
 - The fields that comprise the secondary index must be subordinate fields in the group. You can find the names of these fields in the SRCH parameter of the XDFLD record in the DBD. You described these fields previously in the Master File as sequence or search fields. When you now describe them as subordinate fields, you must assign them a new FIELDNAME not already used in the Master File; their ALIAS values must be the FIELDNAME values you previously gave them.

The following diagram illustrates how to describe a secondary index in the Master File:



1. Is the IMS DBD. It describes one secondary index. The XDFLD record assigns the name IXNAME to the index. The SRCH parameter indicates that the IXNAME index searches on the fields LNAME and FNAME.
2. Is the corresponding Master File. In the GROUP record at the end of the Master File, the ALIAS value is the name of the index from the DBD with the suffix SKY appended.

The subordinate fields in the group were previously assigned fieldnames in the Master File. The ALIAS value for each subordinate field in the group definition is the previously-assigned FIELDNAME. The FIELDNAME in the subordinate field entry is a new name.

The Interface now has the information necessary for determining the best access path for a particular request. Consider the following request:

```

TABLE FILE PATINFO
PRINT LAST_NAME FIRST_NAME SALARY ADMIT_DATE
IF LAST_NAME IS 'SMITH'
END
    
```

Since the field referenced in the IF condition is the high-order part of an index, the Interface generates a qualified SSA to retrieve data using the PCB that permits access through the index. Chapter 4, *Reporting Efficiencies*, shows examples of such SSAs.

Appendix A, *Sample File Descriptions*, illustrates three secondary index definitions for this database.

2.3 Mapping IMS and FOCUS Relationships

The join is an important technique for reporting from any database system. A join enables you to report from more than one database at the same time by matching values of fields from each database.

With most FOCUS Interfaces, you can join separate databases or files that share a common field by describing each file as a segment in one multi-structure Master File. This *embedded join* technique is not available with the IMS/DB Interface since each Master File must describe one PCB and each PCB must describe one DBD.

The following sections describe the techniques available for relating separate IMS databases.

IMS Logical Relationships

In IMS, a PCB can describe a logical database, and a logical DBD can describe a relationship between multiple IMS databases. Therefore, a Master File can describe a join between separate IMS databases if there is a PCB for a DBD that describes the join.

The Interface cannot distinguish between a DBD for a physical database and a DBD for a logical database. You create the Master Files using identical techniques and attributes.

Alternating Between Databases

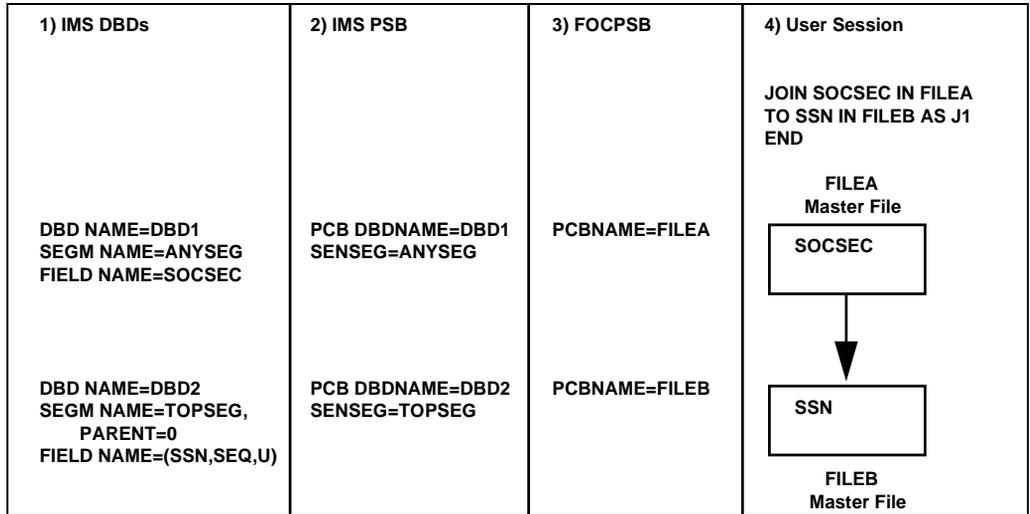
The PSB provides the ability to report from separate IMS databases. The PSB can contain PCBs for multiple databases, and an application can use all PCBs from a PSB. However, this is not a join since there is no automatic matching of common field values from the separate databases and, in fact, the databases do not have to be related in any way.

The Dynamic JOIN

The Interface provides the *dynamic JOIN* command for joining separate databases. You can issue a JOIN command at run time to join IMS databases that have PCBs in the same PSB. In fact, with the appropriate Interfaces installed, you can implement joins between IMS databases and non-IMS databases or files (for example, DB2 tables, MODEL 204 files, and FOCUS databases).

The databases to be joined must have at least one field that shares common values. In order to join “to” an IMS database, you must join to a key field or secondary index in the root segment.

The following diagram illustrates a dynamic join between separate IMS databases:



1. Shows two IMS DBDs. DBD1 has a field called SOCSEC that is not a key and not in the root segment. DBD2 has a field called SSN that shares the same values as SOCSEC. The field SSN is the root key of DBD2.
2. Is a PSB that has PCBs for DBD1 and DBD2. Each PCB is sensitive to the segment containing the shared field.
3. Is the FOCPSB. It associates the first PCB with Master File FILEA and the second PCB with Master File FILEB.
4. Is a FOCUS session or FOCEXEC that issues the JOIN command to join the two Master Files. Notice that the “to” field in the JOIN command is a key in the root segment.

3 Creating FOCUS Descriptions

This chapter explains the rules for making an IMS database accessible to FOCUS. If any of the concepts or terms are unfamiliar to you, you can refer to the discussion of concepts and mapping procedures in Chapter 2, *IMS Overview and Mapping Concepts*.

In IMS, a PSB regulates an application's authority to access databases. The PSB consists of PCBs, each of which describes one view of one database that the program can access. You must describe your PSBs and PCBs to FOCUS before FOCUS can use them to access an IMS database:

- A FOCPSB describes a PSB (see Section 3.1, *The FOCPSB*).
- A Master File describes a PCB (see Section 3.2, *The Master File*).

Each Master File describes the segments and fields that are accessible through its corresponding PCB. The database made accessible by the PCB can be a physical or logical database.

- Optionally, in the DBCTL environment, an Access File can select the appropriate PSB for a request (see Section 3.3, *The Access File*).

In most environments, the user selects a specific PSB at allocation time. However, if you access IMS through the DBCTL environment, you select the PSB during your session, either by issuing a SET command or by creating an Access File to select the PSB.

As discussed in Chapter 2, *IMS Overview and Mapping Concepts*, IMS database descriptions (DBDs) often do not describe every field in the database. However, your site may maintain a COBOL FD description of the database. If it does, you can use the FOCUS COBOL FD Translator to help create Master Files.

A Master File can describe an entire PCB or any part of the PCB that represents a subtree of the hierarchy starting from the root. (See Chapter 2, *IMS Overview and Mapping Concepts*, for a discussion of hierarchies and subtrees.)

When you issue a request such as TABLE FILE SALES, FOCUS processes the request with the following steps:

1. It locates the Master File named SALES.

For MVS, Master Files are stored in an MVS partitioned dataset (PDS) allocated to ddname MASTER; the SALES Master File is member SALES in this dataset.

For CMS, Master Files are stored as files with filetype MASTER; the SALES Master File is the file named SALES MASTER.

2. It examines the SUFFIX attribute in the Master File. Each Master File that describes an IMS database includes the attribute SUFFIX=IMS. When FOCUS detects this SUFFIX, it passes control to the Interface.
3. The Interface examines the user-selected FOCPSB file to identify the PCB that corresponds to the SALES Master File. It uses the information contained in both descriptions to generate the DL/I calls required by the report request. It passes these calls to the IMS DBMS.
4. The Interface retrieves the data generated by the IMS DBMS and returns control to FOCUS. For some requests, FOCUS may perform additional processing on the returned data.

3.1 The FOCPSB

An IMS PSB consists of PCBs. Each PCB represents a view of an IMS database. A FOCPSB describes a PSB to FOCUS and associates a Master File with each PCB in the PSB. Optionally, the FOCPSB can also partition and concatenate PCBs.

In MVS, PSBs are stored as members of a partitioned dataset. FOCPSBs are also stored as members of a partitioned dataset. The member name for a FOCPSB must be identical to the member name of its corresponding PSB.

If your operating system is DOS/VSE, the FOCPSB file must be a sequential VSAM ESDS dataset created with RECFM F.

A FOCPSB is a comma-delimited, free format file that consists of attributes (keyword=value pairs). Rules for declarations are as follows:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,\$). Text appearing after the comma and dollar sign is treated as a comment.
- Certain attributes are required; the rest are optional.

The following sections summarize the syntax for each type of declaration and then present detailed explanations for each attribute.

Note:

- FOCPSBs created in prior FOCUS Releases may consist of fixed format records with no attribute keywords. Appendix C, *Release Dependent Interface Features*, describes fixed format FOCPSBs. While this earlier format is supported, the comma-delimited format is preferable.
- To avoid being enqueued and locked out of a database, the Interface requires its PCBs to specify PROCOPT=GO. With this processing option, you may retrieve a record that is being updated by another user.

3.1.1 Required FOCPSB Attributes

The required FOCPSB attributes describe the PSB to FOCUS and associate a Master File with each PCB you will access.

Section 3.1.2, *Extended FOCPSB Attributes*, describes additional attributes for partitioning and concatenating PCBs.

The Header Record

Each FOCPSB starts with a header record. The syntax is

```
FOCPSB=EXTENDED [ ,PL1=YES] , $
```

where:

FOCPSB=EXTENDED Indicates that the FOCPSB is in comma-delimited format.

PL1=YES Optionally, indicates that the PSB was created for a PL/I application program. You must include this attribute when the IMS PSB specifies LANG=PLI; otherwise, omit it.

Note that you must code the attribute exactly as shown, with the numeric digit 1 in PL1 and the value YES.

The PCB Record

Each PCB in the PSB must have a corresponding record in the FOCPSB. The order of PCB records in the FOCPSB must correspond to the order of the PCBs in the PSB.

If any PCB in the PSB includes the attribute LIST=NO, do not include a corresponding record for that PCB in the FOCPSB.

The syntax for a PCB record in the FOCPSB is

```
PCBNAME=mfdname,PCBTYP=DB [ ,LOWVALUE=val1] [ ,HIGHVALUE=val2],  
PCBNAME=, PCBTYP={TERM|SKIP} [ ,LOWVALUE=val1] [ ,HIGHVALUE=val2],  
$
```

where:

<i>mfdname</i>	Indicates the one- to eight-character name of the Master File for the corresponding database PCB (PCBTYP=DB). To report from the PCB, specify TABLE FILE <i>mfdname</i> .
PCBNAME=	Applies when no Master File is necessary (see TERM or SKIP for details).
TERM	Indicates that the corresponding PCB is an I/O PCB. (You need an I/O PCB to connect to IMS online through a teleprocessing monitor such as IMS/DC.) All I/O PCBs must be listed before any database PCB. Since no Master File is necessary, PCBNAME is blank. Note that if the IMS PSB specifies CMPAT=YES, an I/O PCB is automatically generated at the top of the PSB for batch checkpointing; in this case, you must add an additional I/O PCB at the top of the FOCPSB.
SKIP	Indicates that you will not access the corresponding PCB. Since no Master File is necessary, PCBNAME is blank. Note that SKIP is a reserved word for the FOCPSB.
	Important: Never use SKIP as a Master File name.
<i>val1</i>	Is used for partitioning. See Section 3.1.2, <i>Extended FOCPSB Attributes</i> , for a complete discussion.
<i>val2</i>	Is used for partitioning. See Section 3.1.2, <i>Extended FOCPSB Attributes</i> , for a complete discussion.

A PSB can have duplicate PCBs that provide identical views of an IMS database. Give each of these identical PCBs the same PCBNAME value in the FOCPSB. Certain environments, such as the XMI server environment, use this technique to give multiple users concurrent access to a database (see Chapter 5, *Environments*). You can also use duplicate PCBs to facilitate a recursive join (see Chapter 4, *Reporting Efficiencies*).

The following is an IMS PSB named TSTPSB01. It has two database PCBs that access the PATDB01 database, the first through the primary index, and the second through a secondary index named IXNAME (see Section 3.2.5, *Using a Secondary Index*, for information on describing secondary indexes in the FOCPSB):

```
PCB          TYPE=TP ,MODIFY=YES ,EXPRESS=YES
PCB          TYPE=TP ,EXPRESS=NO ,MODIFY=YES ,SAMETRM=YES
PCB          TYPE=DB ,DBDNAME=PATDB01 ,PROCOPT=GO ,KEYLEN=9
SENSEG      NAME=PATINFO ,PARENT=0
PCB          TYPE=DB ,DBDNAME=PATDB01 ,PROCOPT=GO ,KEYLEN=28 ,PROCSEQ=PATDBIX1
SENSEG      NAME=PATINFO ,PARENT=0
PSBGEN      LANG=COBOL ,PSBNAME=TSTPSB01 ,CMPAT=YES
END
```

The corresponding FOCPSB is shown:

```
FOCPSB=EXTENDED , $
PCBNAME=      ,PCBTYPE=TERM , $
PCBNAME=      ,PCBTYPE=TERM , $
PCBNAME=      ,PCBTYPE=TERM , $
PCBNAME=PATINFO ,PCBTYPE=DB , $
PCBNAME=IXNAME ,PCBTYPE=DB , $
```

3.1.2 Extended FOCPSB Attributes

IMS limits the size of its databases. A site that needs a larger database may be able to create several smaller databases by partitioning the large database based on root key values.

Using extended FOCPSB attributes, you can describe a partition to FOCUS, describe how to concatenate the parts, and assign a name to the concatenated PCB. When you issue a report request, you can report from the concatenated PCB or from any of the individual partitions depending on the filename you reference in the request.

Describing a Partition in the FOCPSB

A partition assigns each record to a specific database depending on its root key value. The first partition contains records with the lowest key values; the next partition contains records with higher key values, and so on; the last partition contains records with the highest key values. Each partition is a separate IMS database and, therefore, has a separate PCB in the PSB. Partitioning is not supported for HDAM databases.

To describe the key range in each partition to FOCUS, add the LOWVALUE and HIGHVALUE attributes to the appropriate PCB records in the FOCPSB. The syntax is

```
PCBNAME=mfdname, PCBTYPEDB, LOWVALUE= {val1|0} ,HIGHVALUE= {val2|FF},$
```

where:

<i>mfdname</i>	Is the name of the Master File for one partition of the large database. (Since the partition is a separate database with its own DBD, it needs its own PCB and Master File.)
<i>val1</i>	Is the lowest key value, in alphanumeric format, in the partition accessed with Master File <i>mfdname</i> . The default is 0.
<i>val2</i>	Is the highest key value, in alphanumeric format, in the partition accessed with Master File <i>mfdname</i> . The default is hexadecimal FF.

Note:

- The LOWVALUE and HIGHVALUE attributes are ignored if you do not define a corresponding concatenation of the PCBs (see the next section).
- The number of partitions you can define for a database is limited by the number of PCBs in the PSB.
- The partitioning field must be the key of the root segment.
- The partitioning field must have an alphanumeric format.
- If the partitioning field is a group composed of multiple sub-fields, each sub-field value must be alphanumeric, and you must specify the concatenated sub-field values in the LOWVALUE and HIGHVALUE attributes. For example, if the root key low value is composed of F1=AAAA, F2=88, and F3=BBB, then LOWVALUE=AAAA88BBB.

The next section shows a sample FOCPSB with partitioning.

Describing a Concatenated PCB in the FOCPSB

In a FOCPSB, you can concatenate individual PCBs by assigning a name to the concatenation and issuing a report request against it.

The PCBs that you concatenate do not have to be partitioned; that is, their FOCPSB records do not have to include the LOWVALUE and HIGHVALUE attributes. However, if they do include the partitioning information, the Interface can examine the report request and determine which PCBs satisfy the request. Without the partitioning information, the Interface must access every PCB that participates in the concatenation.

To concatenate PCBs, include a CONCATNAME record after all PCBNAME records in the FOCPSB. The syntax is

```
CONCATNAME=cname, USE=mfd1/mfd2/.../mfdi , $
```

where:

cname Indicates the concatenation name. You can issue a request from the concatenated PCBs using the syntax TABLE FILE *cname*.

The CONCATNAME record can span more than one line; however, you cannot split an individual Master File name between two lines.

mfd1/.../mfdi Are Master File names from the individual PCBNAME records in the FOCPSB. The key fields for all PCBs you concatenate must be the same length and type. You can issue a request from an individual PCB by referencing its individual Master File name in the request (for example, TABLE FILE *mfd1*).

The following example illustrates how to concatenate partitioned PCBs:

```
FOCPSB=EXTENDED, $
PCBNAME=      , PCBTYP=TERM, $
PCBNAME=      , PCBTYP=TERM, $
PCBNAME=      , PCBTYP=TERM, $
PCBNAME=EMPDB01, PCBTYP=DB, LOWVALUE=000000001, HIGHVALUE=000001667, $
PCBNAME=EMPDB02, PCBTYP=DB, LOWVALUE=000001668, HIGHVALUE=000003334, $
PCBNAME=EMPDB03, PCBTYP=DB, LOWVALUE=000003335, HIGHVALUE=000005000, $
CONCATNAME=EMPDBJ, USE=EMPDB01/EMPDB02/EMPDB03, $
```

Consider the following request (the key field is named SSNALPHA):

```
TABLE FILE EMPDBJ
.
.
.
IF SSNALPHA IS 000001775
.
.
```

Creating FOCUS Descriptions

The Interface satisfies the request using the EMPDB02 PCB only. If the WHERE clause had requested key values less than 000001775 (rather than equal to 000001775), the Interface would have used the EMPDB01 and EMPDB02 PCBs.

Refer to Appendix A, *Sample File Descriptions*, for the corresponding DBD and Master File.

Note:

- If a retrieval request has no WHERE clause, or if the WHERE clause references a non-key field, the Interface accesses all PCBs.
- If any PCB listed in a concatenation lacks the LOWVALUE and HIGHVALUE key range specification, that PCB always participates in the retrieval.

3.1.3 The FOCPSB Dataset

For MVS, each FOCPSB is stored as a member of a partitioned dataset (PDS); by convention, the FOCPSB dataset is named

prefix.FOCPSB.DATA

where:

prefix Is the high-level qualifier used at your site for the FOCUS production libraries.

The member name for a FOCPSB within its PDS must be identical to the member name of the corresponding IMS PSB within its PDS. (IMS PSB library names have the form *prefix1.prefix2.PSBLIB* or *prefix1.prefix2.ACBLIB*.)

In most environments, you select the PSB to use before you invoke the Interface, and you only allocate that FOCPSB member in your CLIST or JCL. However, with the DBCTL environment, you allocate the entire FOCPSB dataset and select a specific PSB during your session. You can implement this selection either dynamically (with a SET command) or by means of an Access File (see Section 3.3, *The Access File*). Chapter 5, *Environments*, discusses allocation of the FOCPSB dataset.

3.1.4 Dynamic PCB Selection Exit

The current release of the IMS Interface has been enhanced to include a new exit IMDYNPCB. This exit allows you to indicate to the Interface which PCB to use for retrieval based on a value in the selection criteria.

The exit is called three times during the execution of a TABLE request:

- At setup time, prior to the TABLE request.

When you issue a TABLE request against an IMS database, the Interface checks to see if a module named IMDYNPCB exists. If it exists, the IMDYNPCB module will be loaded, called and passed the parameters listed in the following section.

You can check the “Call Type” parameter in your code. It is the second parameter in the list. For the setup call, “Call Type” will contain the value “S”.

- At run time, during the TABLE request execution.

Prior to issuing the first DL/I call, the Interface invokes the exit a second time with a “Call Type” value of “R”. Your code should use the information passed to it through the parameters and return a PCB number. This number will be used by the Interface to select the PCB for subsequent data retrieval.

- At termination, at the end of the TABLE request.

After the IMS retrieval is complete (GE), the Interface invokes the exit a third time with a “Call Type” value of “E”. This allows you to do final tasks prior to the Interface shutting down.

How to Use the IMDYNPCB Exit

To use the IMDYNPCB exit you must do the following:

1. Create an Assembler module that contains the following control block. The module must be named IMDYNPCB.

```

DYND      DSECT
EYE       DS    CL7
CALLTYPE  DS    CL1    CALL TYPE can be S-startup, R-runtime, E-terminal
NUMPCBS   DS    F      Number of PCB's in the PSB
PPCBLIST  DS    A      Pointer to PCB list
MFDNAME   DS    CL8    Master File name
KEYNAME   DS    CL8    Key field name
KEYLEN    DS    F      Key length
PKEY      DS    A      Pointer to key value
PCBNUM    DS    F      PCB number to use for data access
          DS    8F
USERAREA  DS    25F

```

2. Place the module in a load library allocated to FOCLIB, USERLIB or STEPLIB.

IMDYNPCB Exit Sample

```

*   Dynamic PCB Selection user routine
*
*   This is a sample only. This code must be modified so that:
*   When called with a calltype of:
*   1) S, any initialization that is needed is processed.
*   2) R, the routine must return in register 15 the pcb number to use based on
*       the key value passed in pkey. The first pcb in the psb is 1, not 0.
*   3) E, any shutdown code is processed. Any system resources allocated in
*       startup must be freed here.
*   In this example, a 9 byte key is examined and the pcb number to use
*   is returned
*
      PRINT GEN
IMDYNPCB startup code goes here (save regs in savearea (reg 13) )
      L     R3,0(R1)
      USING DYND,R3
*
      XR   R7,R7           initialize return code
*
STARTUP  CLI   CALLTYPE,C'S'
        BNE  RUNTIME
*
*       PLACE STARTUP CODE HERE
*
      B     IMDEND
*
RUNTIME  CLI   CALLTYPE,C'R'
        BNE  IMDCLOSE
*
*       TEST KEY VALUE PASSED AND SET R7 (WHICH WILL BE RETURNED)
*       TO THE PCB NUMBER YOU WANT TO USE
*
      L     R4,PKEY
      LA   R7,22
      CLC  0(9,R4),=C'000001667'
      BL  IMDEND
      LA   R7,23
      CLC  0(9,R4),=C'000003334'
      BL  IMDEND
      LA   R7,24
      B     IMDEND
*
IMDCLOSE CLI   CALLTYPE,C'E'
        BNE  IMDEND
*
*       PLACE SHUTDOWN CODE HERE
*
IMDEND  return, restore registers, set R15 to R7 and branch to R14
*
      LTORG
DYND    DSECT
EYE    DS    CL7
CALLTYPE DS  CL1    CALL TYPE can be S-startup, R-runtime, E-terminal
NUMPCBS DS  F      Number of PCB's in the PSB
PPCBLIST DS  A      Pointer to PCB list
MFDNAME DS  CL8    Master File name
KEYNAME DS  CL8    Key field name
KEYLEN  DS  F      Key length
PKEY    DS  A      Pointer to key value
PCBNUM  DS  F      PCB number to use for data access
        DS    8F
USERAREA DS  25F
        END

```

3.2 The Master File

Each Master File that provides access to an IMS database describes the segments and fields that are available through one IMS PCB.

You do not have to describe every segment from the PCB in the Master File. However, the portion of the hierarchy you describe must be a subtree starting from the root (see Chapter 2, *IMS Overview and Mapping Concepts*). Any segment or field that you do not describe in the Master File remains invisible to FOCUS.

Reporting costs are largely a function of the volume of data transferred. Therefore, requests issued through the Interface are efficient and cost effective, because only segments referenced in your request are retrieved.

In a Master File, you can describe up to 64 segments across 15 levels. The cumulative length of all fields across all segments cannot exceed 12,000 bytes (an IMS restriction. Section 3.2.3, *Field Attributes*, contains additional information about this limit).

For MVS, each Master File is stored as a member of a Master File PDS. The member name for a Master File must be the name assigned to that Master File in the FOCPSB record for the corresponding PCB (see *The PCB Record* in Section 3.1.1, *Required FOCPSB Attributes*). At run time, the Master File dataset is allocated to ddname MASTER.

For CMS, each Master File is stored as a file with filetype MASTER. The filename must be the name assigned in the corresponding FOCPSB record.

A Master File consists of file, segment, and field declarations. Rules for declarations are as follows:

- Each declaration must begin on a separate line and be terminated by a comma and dollar sign (,\$). Text appearing after the comma and dollar sign is treated as a comment.
- A declaration can span as many lines as necessary as long as no attribute=value pair is separated. Commas are used to separate attribute=value pairs.

The following sections summarize the syntax for each type of declaration and then describe each attribute.

3.2.1 File Attributes

Each Master File begins with a file declaration that names the file and describes the type of data source—an IMS database in this case. The file declaration has two attributes, FILENAME and SUFFIX. The syntax is

```
FILE[NAME]=filename, SUFFIX=IMS [, $]
```

where:

<i>filename</i>	Is any 1- to 8-character file name.
SUFFIX=IMS	Indicates that the IMS/DB Interface is required for data retrieval.

3.2.2 Segment Attributes

Each IMS segment described in a Master File requires a segment declaration that consists of at least two attributes, SEGNAME and SEGTYPE. The SEGNAME value is the name of the corresponding IMS segment; the SEGTYPE value identifies the segment's characteristics.

The syntax is

```
SEGNAME=segname ,SEGTYPE= segtype [, PARENT=parent] [, $]
```

where:

<i>segname</i>	Is the one- to eight-character IMS segment name from the NAME parameter of the SENSEG record in the IMS PCB (see Chapter 2, <i>IMS Overview and Mapping Concepts</i>).
<i>segtype</i>	Identifies the characteristics of the segment. Possible values are as follows: <ul style="list-style-type: none"> U indicates that the segment is data sensitive and unique. A unique segment has no twins (PTR=NOTWIN in the IMS DBD). While a parent segment can have many <i>types</i> of unique children, it can have at most one <i>instance</i> of each type. S0 indicates that the segment is data sensitive and has no key. S or S1 indicates that the segment is data sensitive and has a non-unique key. S2 indicates that the segment is data sensitive and has a unique key. SH or SH1 indicates that the segment is key sensitive and has a non-unique key. SH2 indicates that the segment is key sensitive and has a unique key.
<i>parent</i>	Is the name of the parent segment from the IMS database. Its value comes from the PARENT parameter of the SENSEG record in the IMS PCB.

SEGNAME

The SEGNAME attribute identifies the IMS segments you can access. The SEGNAME value is the name of the IMS segment from the SENSEG record in the PCB.

FOCUS retrieves segments in top-to-bottom, left-to-right sequence as described by the Master File. Therefore, the order of segments in the Master File should be the same as their order in the PCB to maintain the correct hierarchical sequence. However, FOCUS retrieves unique segments (only one instance per parent—no twins) prior to any non-unique segments that have the same parent.

SEGTYPE

The SEGTYPE attribute does the following:

- Identifies the characteristics of the segment's key field. A segment can have a unique key, a non-unique key, or no key.

If a segment has a unique key, at least one FIELD record in the DBD must specify NAME=(fieldname,SEQ,U) or NAME=(fieldname,SEQ); similarly, if the segment has a non-unique key, at least one FIELD record in the DBD must specify NAME=(fieldname,SEQ,M). If no FIELD record in the DBD specifies a SEQ attribute, the segment has no key. The key referred to by the SEGTYPE attribute may actually be a secondary index (see Section 3.2.5, *Using a Secondary Index*).

The characteristics of the segment's key determine the types of SSAs and the number of DL/I calls the Interface must issue to satisfy a retrieval request. To handle keys made up of multiple fields, describe the multiple fields as a group in the Master File (see Section 3.2.4, *Group Fields*). If you do not specify a key field (.KEY) for a keyed SEGTYPE (for example, S1), a FOC4277 message will result.

- Identifies unique segments. A segment is unique if it has no twins. A segment can be parent to at most one instance of a unique segment type. The SEGM record in the DBD for a unique segment specifies PTR=NOTWIN.
- Identifies whether the data from the segment can be retrieved (data sensitive) or if only the segment's key is accessible (key sensitive). For a key sensitive segment, the SENSEG record in the PCB includes the parameter PROCOPT=K. Key sensitive segments are used for access to lower level segments.

The following table lists the valid SEGTYPE values:

SEGTYPE	Definition	PROCOPT=K In PCB?	NAME= From DBD
U	Data sensitive, unique segment	No	Not Applicable
S0	Data sensitive, no key	No	(name)
S or S1	Data sensitive, non-unique key	No	(name,SEQ,M)
S2	Data sensitive, unique key	No	(name,SEQ,U) or (name,SEQ)
SH or SH1	Key sensitive, non-unique key	Yes	(name,SEQ,M)
SH2	Key sensitive, unique key	Yes	(name,SEQ,U) or (name,SEQ)

PARENT

The PARENT attribute identifies the segment's parent in the hierarchy. It appears in the PARENT parameter of the SENSEG record in the PCB. The only exception is in the root segment, where the PCB either omits the PARENT parameter or specifies PARENT=0. In the Master File, you can specify the PARENT attribute of the root segment as PARENT=, or you can omit it.

3.2.3 Field Attributes

Each segment consists of one or more fields. The IMS DBD contains FIELD declarations for all sequence (key) and search fields, but other fields are optional.

If the PCB you are describing contains SENFLD records for a segment, the Master File can view only fields explicitly specified in those SENFLD records.

However, if the PCB does not contain any SENFLD records for a segment, you can describe the entire segment in the Master File. You can get information about sequence and search fields from the DBD; to describe other fields you may have to refer to the COBOL FD description.

The Master File must describe the entire IMS segment. If it does not need access to some of the fields from a segment, it can replace their definitions with filler fields that occupy the same amount of space.

To describe a field in the Master File, you must supply the primary attributes FIELDNAME, ALIAS, USAGE, and ACTUAL, discussed in this section.

The *FOCUS for IBM Mainframe Users Manual* explains additional attributes, such as TITLE and DESCRIPTION, as well as how to define temporary fields in the Master File.

Note: The Interface does not support the MISSING attribute.

The syntax for a field declaration is

```
FIELD[NAME]=fieldname, [ALIAS=alias], [USAGE=]display, [ACTUAL=]imsformat, $
```

where:

fieldname Is a 1- to 66-character fieldname. In requests, you can qualify the fieldname with the Master File and/or segment name; although the qualifiers and qualification characters do not appear in the Master File, they count toward the 66-character maximum.

alias Is an alias for an IMS field. Possible values are as follows:

imsname.KEY is the alias for an IMS key field (except for the root key of an HDAM database). Form the alias by appending the suffix KEY to the name of the IMS field.

imsname.IMS is the alias for an IMS search field. Form the alias by appending the suffix IMS to the name of the IMS field.

imsname.HKY is the alias for the key of the root segment in an HDAM database. Form the alias by appending the suffix HKY to the name of the IMS field.

anyname is the alias for a field not listed in the DBD. The alias is any name, up to 66 characters, that complies with FOCUS field naming conventions. The alias value can also be blank.

display Is the FOCUS display format for the field.

imsformat Is the FOCUS definition of the IMS field format and length (n).

You can omit the ALIAS, USAGE, and ACTUAL keywords from the field declaration if the values are specified in the standard order (FIELD, ALIAS, USAGE, ACTUAL). For example, the following declarations are equivalent:

```
FIELD = YEAR, ALIAS=YR, USAGE=A2, ACTUAL=A2, $
FIELD = YEAR, YR, A2, A2, $
```

FIELDNAME

Fieldnames can have up to 66 alphanumeric characters. IMS field names are acceptable values if they meet the following naming conventions:

- Names can consist of letters, digits, and underscore characters. Special characters and embedded blanks are not advised.
- The name must contain at least one letter.

Since fieldnames appear as default column titles for reports, select names that are representative of the data.

You can use fieldnames, aliases, or a unique truncation of either in requests. For example, consider the following Master File:

```
FILENAME=MFD1
SEGNAME=SEG1
FIELDNAME=S1, ALIAS=K1.KEY, USAGE=A4, ACTUAL=A4,$
```

The following statements are all equivalent in a FOCUS request:

```
PRINT S1
PRINT S
PRINT K1.KEY
PRINT K1
PRINT K
```

Duplicate fieldnames (the same fieldnames and aliases) within a segment are not permitted. Report requests can qualify duplicate fieldnames from *different* segments with the name of the Master File and/or segment. That is, using the Master File from the previous example, the following are equivalent:

```
PRINT SEG1.S1
PRINT MFD1.S1
PRINT MFD1.SEG1.S1
```

You need this type of qualification if two segments in the Master File or in a join have a field defined with the same fieldname and alias. For more information about field qualifiers, see the *FOCUS for IBM Mainframe Users Manual*.

Note:

- The maximum length for fieldnames and aliases is 66 characters. This is the result of the default setting (NEW) for the FOCUS SET FIELDNAME command. If the setting is changed to OLD, the maximum length changes to 12 characters. A third setting, NOTRUNC, prevents unique truncations of fieldnames and supports the 66-character maximum.

- If you change the setting to OLD or vice versa, existing dynamic joins and DEFINE fields are released. If you change the setting to NOTRUNC, existing joins and DEFINE fields are unaffected. For more information about long and qualified fieldnames, see the *FOCUS for IBM Mainframe Users Manual*.
- Although field qualifiers and qualification characters are used only in requests and cannot appear in the Master File, they count toward the maximum of 66 characters allowed for fieldnames. Therefore, if you define a 66-character fieldname in the Master File, you cannot qualify the fieldname in a request.
- Fieldnames in OCCURS segments may not be qualified.

ALIAS

The ALIAS value in the Master File distinguishes between fields that are defined in the IMS DBD and fields that are not defined to IMS. The Interface uses this information in constructing DL/I calls to IMS.

If a field used in a screening test (IF or WHERE) is a sequence or search field, the Interface may be able to create an SSA that instructs IMS to apply the screening test and return the appropriate records to FOCUS; if the field is not defined in the DBD, the Interface must retrieve all records sequentially from IMS so that FOCUS can screen them.

In certain cases, the Interface can instruct IMS to screen values based on a secondary index. Section 3.2.5, *Using a Secondary Index*, describes the technique for taking advantage of a secondary index.

The ALIAS value for a field defined in the DBD is composed of the following:

- The name of the field as specified in the IMS DBD.
- A separation character, the period (.).
- A suffix value that describes whether the field is a sequence field (suffix KEY), a search field (suffix IMS), or the key of the root segment of an HDAM database (suffix HKY).

You can assign fields not defined in the DBD any name that complies with FOCUS field naming conventions. Do not include periods or suffix values for such alias names.

In requests, you can reference a field by its fieldname, its alias, or by a unique truncation of either. Note that the IMS fieldname is a unique truncation of the alias value. For example, if the IMS key field is named IMSNAME, then the alias for that field in the Master File is IMSNAME.KEY. A FOCUS request can reference the field by the name IMSNAME (without the .KEY) because this name is a unique truncation of the alias.

USAGE

The USAGE attribute indicates the display format of the field. An acceptable value must include the field type and length and may contain edit options. FOCUS uses the USAGE format for data display on reports. All standard FOCUS USAGE formats (A, D, F, I, P) are available. For additional information about USAGE formats for external files, see the *FOCUS for IBM Mainframe Users Manual*.

ACTUAL

The ACTUAL attribute indicates the FOCUS representation of IMS field formats.

For fields defined in the DBD (sequence and search fields), use the format specified in the DBD.

Use the following chart as a guide for describing ACTUAL formats of those fields *not* defined in the DBD:

COBOL FORMAT	COBOL PICTURE	BYTES OF STORAGE	ACTUAL FORMAT	USAGE FORMAT
DISPLAY	X(4)	4	A4	A4
DISPLAY	S99	2	Z2	P3
DISPLAY	9(5)V9	6	Z6.1	P8.1
DISPLAY	99	2	A2	A2
COMP	S9	4	I2	I1
COMP	S9(4)	4	I2	I4
COMP	S9(5)	4	I4	I5
COMP	S9(9)	4	I4	I9
COMP-1	-	4	F4	F6
COMP-2	-	8	D8	D15
COMP-3	9	1	P1	P1
COMP-3	S9V99	2	P2	P5.2
COMP-3	9(4)V9(3)	4	P4	P8.3
FIXED BINARY(7) (COMP-4)	B or XL1	4	I4	I7

The USAGE lengths shown are minimum values. You can make them larger and add edit options. You must allow space for all possible digits, a minus sign for negative numbers, and a decimal point in numbers with decimal digits. For more information, see the *FOCUS for IBM Mainframe Users Manual*.

The cumulative length of all fields, across all segments, cannot exceed 12K bytes. To determine whether the database you are defining falls within these constraints, issue the following report request:

```
CHECK FILE filename HOLD
TABLE FILE HOLD
WRITE MAX.LWORD
END
```

The resulting value must be less than or equal to 3,000.

The following example illustrates the DI21PART Master File that provides access to the DI21PART database.

```
FILE=DI21PART      , SUFFIX=IMS , $
SEGNAME=PARTROOT  , PARENT=    , SEGTYPE=S2 , $
  FIELD=PARTKEY    , ALIAS=PARTKEY.HKY , USAGE=A17 , ACTUAL=A17 , $
  FIELD=SKIP1      , ALIAS=SKIP1      , USAGE=A33 , ACTUAL=A33 , $
SEGNAME=STANINFO  , PARENT=PARTROOT , SEGTYPE=S2 , $
  FIELD=STANKEY    , ALIAS=STANKEY.KEY , USAGE=A2   , ACTUAL=A2   , $
  FIELD=SKIP2      , ALIAS=SKIP2      , USAGE=A83 , ACTUAL=A83 , $
SEGNAME=STOKSTAT  , PARENT=PARTROOT , SEGTYPE=S2 , $
  FIELD=STOCKEY    , ALIAS=STOCKEY.KEY , USAGE=A16 , ACTUAL=A16 , $
  FIELD=SKIP3      , ALIAS=    , SKIP3      , USAGE=A124 , ACTUAL=A124 , $
SEGNAME=CYCCOUNT  , PARENT=STOKSTAT , SEGTYPE=S2 , $
  FIELD=CYCKEY     , ALIAS=CYCKEY.KEY , USAGE=A2   , ACTUAL=A2   , $
  FIELD=SKIP4      , ALIAS=SKIP4      , USAGE=A23 , ACTUAL=A23 , $
SEGNAME=BACKORDR  , PARENT=STOKSTAT , SEGTYPE=S2 , $
  FIELD=BACKEY     , ALIAS=BACKEY.KEY , USAGE=A10 , ACTUAL=A10 , $
  FIELD=SKIP5      , ALIAS=SKIP5      , USAGE=A65 , ACTUAL=A65 , $
```

3.2.4 Group Fields

IMS fields can consist of multiple elementary fields. In the Master File, you can break the IMS field into component parts using a GROUP field.

The GROUP record in the Master File describes the combined elementary fields. FIELD records immediately following the GROUP record describe the individual elementary fields.

Each element of the group can have a different format. However, retrieval is more efficient if each element's USAGE format is the same type as its ACTUAL format (for example, alphanumeric or packed); the lengths may differ.

Creating FOCUS Descriptions

The syntax for a GROUP record and its subordinate FIELD records is

```
GROUP=gname, ALIAS=alias, USAGE=An, ACTUAL=Am, $
FIELD=fld1, all, usel, act1, $
.
.
.
FIELD=fldn, aln, usen, actn, $
```

where:

<i>gname</i>	Is the group name. It can be any name that complies with FOCUS field naming conventions.										
<i>alias</i>	Is an alias for an IMS field. Note that the keyword ALIAS is required. Possible values are as follows: <i>imsname</i> .KEY is the alias for an IMS key field (except for the root key of an HDAM database). Append the suffix KEY to the name of the IMS field. <i>imsname</i> .IMS is the alias for an IMS search field. Append the suffix IMS to the name of the IMS field. <i>imsname</i> .HKY is the alias for the key of the root segment in an HDAM database. Append the suffix HKY to the name of the IMS field. <i>anyname</i> is the alias for a field not listed in the DBD. The alias is any name, up to 66 characters, that complies with FOCUS field naming conventions. The alias value can also be blank.										
<i>An</i>	Is the USAGE format for the GROUP. It must be alphanumeric and its length must count all F fields as length 4, all D fields as length 8, and all integer fields as length 4, regardless of the length specified in the ACTUAL format. For alphanumeric fields, USAGE length must equal ACTUAL length. For P (packed) fields, the length depends on the USAGE of the packed field: <table> <thead> <tr> <th>USAGE of Packed Field</th> <th>Length for GROUP USAGE</th> </tr> </thead> <tbody> <tr> <td>Pn or Pn.d, n≤15, P16</td> <td>8</td> </tr> <tr> <td>P16</td> <td>8</td> </tr> <tr> <td>P16.d</td> <td>16</td> </tr> <tr> <td>Pn or Pn.d, 16<n≤31</td> <td>16</td> </tr> </tbody> </table>	USAGE of Packed Field	Length for GROUP USAGE	Pn or Pn.d, n≤15, P16	8	P16	8	P16.d	16	Pn or Pn.d, 16<n≤31	16
USAGE of Packed Field	Length for GROUP USAGE										
Pn or Pn.d, n≤15, P16	8										
P16	8										
P16.d	16										
Pn or Pn.d, 16<n≤31	16										
<i>Am</i>	Is the ACTUAL format for the GROUP. Its length is the sum of the lengths of the IMS fields.										
<i>fld1</i> , ..., <i>fldn</i>	Are FOCUS field names for the individual elements in the group.										
<i>all</i> , ..., <i>aln</i>	Are alias names for the individual elements. They can be any names that comply with FOCUS field naming conventions.										

- use1, ..., usen* Are USAGE formats for the individual elements. For efficient retrieval, each individual USAGE format must be of the same data type as its corresponding ACTUAL format, but their lengths can differ.
- act1, ..., actn* Are ACTUAL formats for the individual elements. For efficient retrieval, each individual USAGE format must be of the same data type as its corresponding ACTUAL format, but their lengths can differ.

The following is an example of a group key definition in the Master File:

```
GROUP=G1, ALIAS=FILEKEY.KEY, USAGE=A16, ACTUAL=A11 , $
FIELD=F1, F1, A4, A4, $
FIELD=F2, F2, P6, P3, $
FIELD=F3, F3, I9, I4, $
```

Note:

- If the key to an IMS segment is composed of multiple fields, you must define the key as a GROUP. Only one field in a segment can have the .KEY suffix; for a group key, this field must be the GROUP field.
- The GROUP in the example describes an IMS key named FILEKEY that is 11 bytes long and consists of a 4-byte alphanumeric field, a 3-byte packed number, and a 4-byte integer.
- The ACTUAL length of the GROUP is 11 (4+3+4).
- The USAGE length of the GROUP is 16 (4+8+4), because it counts the packed field as 8.

You can use the group field or any individual field in a FOCUS query. For example:

```
WHERE G1 EQ 'ABCD'/245/-20 or
IF G1 GT ABCD/245/-20
IF G1 FROM ABCD/245/-20 TO ABCD/300/50
IF G1 IS ABCD/245F/-20 (field F2 has a non-standard sign, F)
IF G1 IS A$*
IF F1 IS ABCD
IF F1 FROM A TO B
IF F2 IS 245
```

You must use a slash (/) to separate the individual fields when you reference the group name.

If you reference either the group or the first elementary field from the group in your request, the Interface generates qualified SSAs in its DL/I calls for retrieval. Thus, IMS does the screening and returns the segments that pass the screening test back to FOCUS.

However, if you reference an elementary field that is not the first field in the group, FOCUS constructs DL/I calls to retrieve the segments sequentially, and then FOCUS applies the screening test to the returned data.

Chapter 4, *Reporting Efficiencies*, contains a detailed explanation of screening conditions and SSA generation.

Note: The Interface does not support a group field within a group field. See the *FOCUS for IBM Mainframe Users Manual* for information on DEFINE fields in the Master File.

3.2.5 Using a Secondary Index

Using IMS secondary indexes, you can retrieve records in order of a field other than the key field. (A secondary index is itself a database with its own DBD.) The DBD for a database that uses a secondary index includes an XDFLD statement that assigns a field name to the index.

If a PCB includes the parameter PROCSEQ=indexDBDname, the named index is used as the main entry point into the database.

In prior FOCUS Releases, each Master File could describe only one index and the application programmer had to decide which Master File provided the optimal access path for a request. This technique is still supported, and Appendix C, *Release Dependent Interface Features*, describes it.

Starting with FOCUS Release 7.0, one Master File and FOCPSB can describe all primary and secondary indexes for a database. Then, given a report request, the Interface can examine all record selection tests to determine the best access path into the database. The Interface can take advantage of this Auto Index Selection feature for the following reasons:

- The PSB includes a PCB for each index you may need to access. Each such index PCB must contain a PROCSEQ=indexDBDname parameter that identifies the index DBD.
- The index targets the root segment in the Master File.

In the Master File, prior to the secondary index definitions, you must describe the entire root segment of the database. Every field listed in the DBD is an IMS sequence field or search field, and each secondary index is based on one or more of these fields. You must assign each secondary index field its appropriate alias, as described in *ALIAS* in Section 3.2.3, *Field Attributes*.

Note: IMS allows for system-defined sub-sequence fields to make an index unique. The Master File can completely ignore the presence and length of such fields.

For each secondary index, in the Master File you must perform the following steps:

1. Describe the secondary index as a group field *at the end of the root segment*. In the group definition, use the index name (provided by the XDFLD NAME parameter in the DBD) as the alias name and append the suffix SKY to it. The syntax is

```
GROUP=anyname, ALIAS=XDFLDname. SKY
```

where:

<i>anyname</i>	Is the fieldname for the group, unique within the Master File.
<i>XDFLDname</i>	Is the name assigned to the index by the XDFLD record in the DBD.

For example, consider the following portion of the DBD for the PATDB01 database:

```

.
.
.
FIELD NAME=LNAME, BYTES=12, START=56, TYPE=C
FIELD NAME=FNAME, BYTES=12, START=68, TYPE=C
.
.
.
LCHILD NAME=(SEGIX1, PATDBIX1), PTR=INDX
XDFLD NAME=IXNAME, SRCH=(LNAME, FNAME), X
      SUBSEQ=/SX1, NULLVAL=BLANK

```

In the example, the fields that comprise the secondary index are LNAME, FNAME, and an additional sub-sequence system field that makes the index unique and that the Master File can ignore. The GROUP definition for the index in the Master File follows:

```
GROUP=NAMEIX , ALIAS=IXNAME.SKY , USAGE=A24 , ACTUAL=A24, $
```

The ALIAS is the index name from the DBD (XDFLD NAME=IXNAME) with the suffix SKY appended. Note that the index definition completely ignores the sub-sequence field.

2. Describe the indexed fields as subordinate fields of the group. The XDFLD SRCH parameter in the DBD lists the names of the fields that participate in the index. You already described these fields once in the Master File as sequence or search fields.

Assign each subordinate field a new fieldname not previously used in the Master File. Give each subordinate field an alias value identical to the fieldname you assigned it when you previously described it as a sequence or search field. The syntax is

```

FIELDNAME=fieldname, ALIAS=alias
.
.
.
GROUP=...
FIELD[NAME] = newname, ALIAS = fieldname

```

where:

<i>alias</i>	Is an alias for an IMS field. Possible values are as follows: <i>imsname</i> .KEY is the alias for an IMS key field (except for the root key of an HDAM database). Append the suffix KEY to the name of the IMS field. <i>imsname</i> .IMS is the alias for an IMS search field. Append the suffix IMS to the name of the IMS field. <i>imsname</i> .HKY is the alias for the key of the root segment in an HDAM database. Append the suffix HKY to the name of the IMS field.
<i>newname</i>	Is any name not previously used in the Master File.
<i>fieldname</i>	Is the fieldname previously assigned to this field.

Creating FOCUS Descriptions

The following portion of the PATINFO Master File illustrates the secondary index definition for the example in Step 1:

```
      .
      .
      .
FIELD=LAST_NAME      , ALIAS=LNAME . IMS      , USAGE=A12      , ACTUAL=A12 , $
FIELD=FIRST_NAME    , ALIAS=FNAME . IMS      , USAGE=A12      , ACTUAL=A12 , $
      .
      .
      .
GROUP=NAMEIX        , ALIAS=IXNAME . SKY      , USAGE=A24 ,      , ACTUAL=A24 , $
FIELD=NAMEL        , ALIAS=LAST_NAME      , USAGE=A12      , ACTUAL=A12 , $
FIELD=NAMEF        , ALIAS=FIRST_NAME      , USAGE=A12      , ACTUAL=A12 , $
```

The FOCPSB must also reflect the secondary indexes. The IMS PSB includes a PCB for the normal entry point into the database and an additional PCB for entry through each secondary index. Each PCB for a secondary index includes the parameter PROCSEQ=indexDBDname, where indexDBDname comes from the LCHILD NAME parameter in the database DBD.

The FOCPSB must have a one-to-one correspondence with the PSB.

- The FOCPSB entry that corresponds to the PCB for the normal entry point into the database must identify the name of the Master File. This example illustrates the PCB for the normal entry point into the PATDB01 database and its corresponding FOCPSB entry.

PCB:

```
PCB          TYPE=DB, DBDNAME=PATDB01, PROCOPT=GO, KEYLEN=9
SENSEG      NAME=PATINFO, PARENT=0
```

FOCPSB:

```
PCBNAME=PATINFO, PCBTYPE=DB, $
```

- Any FOCPSB entry that corresponds to a secondary index PCB must identify the name of the index. This index name is the ALIAS of the GROUP record for the index in the Master File. It is also the value of the XDFLD NAME parameter in the DBD.

The next example illustrates a secondary index PCB and its corresponding FOCPSB entry for the PATDB01 database.

PCB:

```
PCB          TYPE=DB, DBDNAME=PATDB01, PROCOPT=GO, KEYLEN=9, PROCSEQ=PATDBIX1
SENSEG      NAME=PATINFO, PARENT=0
```

FOCPSB:

```
PCBNAME=IXNAME, PCBTYPE=DB, $
```

When the Interface generates DL/I calls for retrieval, it examines the record selection tests in the request to determine which PCB offers the most efficient access path to the required data. See Chapter 4, *Reporting Efficiencies*, for information.

Note: Since the PSB most likely includes only one PCB for each secondary index, each Master File that accesses the same index PCB must contain the same GROUP ALIAS value for the index.

Appendix C, *Release Dependent Interface Features*, includes a sample DBD, PSB, FOCPSB, and Master File for the PATDB01 database and its three secondary indexes.

3.2.6 Segment Redefinition: The RECTYPE Attribute

An IMS segment can have multiple definitions. For instance, a segment may contain either shipment or order information, depending on the value of one of its fields. If the field that identifies the type of segment is at the same position and has the same format and length in each redefinition, you can use the RECTYPE attribute to define the different segment types in the Master File.

The record type (RECTYPE) field can be part of the key or of the body of the segment. When you issue a request, you do not have to know which segment definition is called for. FOCUS displays the appropriate fields and values based on the value in the RECTYPE field.

In the Master File, you describe the one IMS segment with multiple FOCUS segments: a base segment describing the unchanging portion, and a child segment describing each redefinition:

1. First define the constant portion of the segment as the base segment; give it the same name as the IMS segment. Include a filler field for the portion that will be redefined. The field that identifies the different types must be in the redefined portion.
2. Describe each redefined portion as a child of the base segment. You can give these children any valid segment names, but do not assign them SEGTYPE values. Include a filler field in each child to occupy the positions of fields actually defined in the base segment.
3. In each child segment, describe the field that identifies the segment type with FIELDNAME=RECTYPE. The value in the RECTYPE field is the value that identifies the type of segment; for example, the RECTYPE field could contain the value S for a shipment record, O for an order record. The format of the RECTYPE field can be alphanumeric, integer, or packed.
4. Assign the identifying value (for example, S or O) as the ALIAS for the RECTYPE field. If more than one value can identify the same record type, use the ACCEPT attribute, discussed in the next section, instead.
5. Describe the remaining fields of each child segment based on its contents and function.

Creating FOCUS Descriptions

The following example illustrates an IMS DBD with a redefined segment. The CLIENT segment contains client id, address, and other client information. The INFO segment contains either shipment information or order information, depending on the value in the INFOTYPE field. If INFOTYPE contains the value S, the segment is a shipment segment; if INFOTYPE contains the value O, the segment is an order segment. The relevant portions of the DBD are shown:

```
SEGM NAME=CLIENT, BYTES=( 200 ), PTR=( TWIN ), PARENT=0
  FIELD=( CLID, SEQ, U ), BYTES=8, START=1, TYPE=C
  .
  .
  .
SEGM NAME=INFO, BYTES=( 200 ), PTR=( TWIN ), PARENT=CLIENT
  FIELD=( IKEY, SEQ, U ), BYTES=8, START=1, TYPE=C
  FIELD=( INFOTYPE ), BYTES=1, START=09, TYPE=C
  .
  .
  .
```

The corresponding Master File represents the IMS INFO segment with three segments:

```
FILE=IMS1, SUFFIX=IMS
  SEGNAME=CLIENT, SEGTYPE=S2
    FIELD=F1, CLID.KEY, A8 ,A8 , $
    FIELD=F2, CLNAME , A20, A20, $
1.  SEGNAME=INFO , SEGTYPE=S2, PARENT=CLIENT, $
    FIELD=F3, IKEY.KEY, A8 ,A8 , $
    FIELD=, , A20, A20, $
2.  SEGNAME=SHIP , SEGTYPE=, PARENT=INFO, $
    FIELD=, , A8 ,A8 , $
    FIELD=RECTYPE ,S, A1 ,A1 , $
    FIELD=SHIPDATE, , A6 ,A6 , $
    .
    .
    .
    other shipment info
3.  SEGNAME=ORDER, SEGTYPE=, PARENT=INFO, $
    FIELD=, , A8 ,A8 , $
    FIELD=RECTYPE, O, A1 ,A1 , $
    FIELD=ORDERDATE, , A6 ,A6 , $
    .
    .
    .
    other order info.
```

1. The base segment, like the IMS segment, is named INFO. It contains the key field; the redefined portion is described as a filler field.
2. The SHIP segment describes the shipment record type; the field that corresponds to the IMS INFOTYPE field has FIELDNAME=RECTYPE and ALIAS=S.
3. The ORDER segment describes the order record type; the field that corresponds to the IMS INFOTYPE field has FIELDNAME=RECTYPE and ALIAS=O.

Notice that each child segment has a filler for the key field defined in the base segment.

Accepting Multiple RECTYPE Values

If multiple values identify the same record type (for example, S or T for a shipment record), use the ACCEPT attribute to enumerate the list or range of acceptable values. In this case, define the ALIAS value as blank.

The syntax is

```
FIELDNAME=RECTYPE, ALIAS= , USAGE=usage, ACTUAL=actual,
    ACCEPT = val1 [OR] val2 [ [OR] ...valn] , $
    ACCEPT = val1 to val2 , $
```

where:

<i>usage</i>	Is the FOCUS display format for the field.
<i>actual</i>	Is the FOCUS definition of the IMS field format and length (n).
<i>val1, val2, valn</i>	Defines a list or range of values that identifies the record type. A list can be continued on more than one line. Enclose values that contain embedded blanks or special characters in single quotation marks.

The following examples illustrate the ACCEPT attribute:

```
ACCEPT = AAA TO RRR
ACCEPT = 136 TO 1029
ACCEPT = RED OR WHITE OR BLUE
ACCEPT = RED WHITE BLUE
ACCEPT = 6 OR 11 OR 922 OR 1000
ACCEPT = RED WHITE 'GREEN GREY'
```

Refer to the Logical Parent Segment feature in Chapter 2, *IMS Overview and Mapping Concepts*, for more information on RECTYPES.

3.2.7 Variable Length Segments: The OCCURS Segment

In an IMS database, segments can have repeating fields or repeating groups of fields. The number of repetitions may have the following attributes:

- It can be a fixed number.
- It can depend on the value of a field from the parent segment or from the non-repeating portion of the variable segment.
- It may have to be calculated from the segment length.

Creating FOCUS Descriptions

In the Master File, you define multiple segments to describe one IMS variable length segment:

- Define the fixed (single-occurrence) portion of the IMS segment as the base segment; give it the same name as the IMS segment.
- Define each repeating field or group of fields from the IMS segment as a child segment whose parent is the base segment. The child segment definition has no SEGTYPE, but it includes the OCCURS attribute to specify how many times the field repeats.
- Define the ORDER field in the OCCURS segment if you need to associate a sequence number with each occurrence.

The OCCURS segment is a virtual segment (it does not physically exist) that describes the repetitions to FOCUS. The following chart lists permissible values for the OCCURS attribute:

OCCURS=	Description
n	Is the number of times the field repeats in the segment.
fieldname	Is the name of a field that contains a value indicating the number of times the field repeats in the segment. The repeating field must be at the end of the segment.
VARIABLE	Indicates that the number of repetitions must be computed from the length of the segment. In this case, the segment must contain a counter field as its first field; the counter field alias in the Master File must be IMSname.CNT. The repeating field must be at the end of the segment.

When the number of occurrences is fixed, the repeating field can be located between two fields rather than at the end of the segment. In this case, you must define a place-holder field at the repeating field's position in the base segment. Then, in the OCCURS segment, you must indicate the location of the repeating field by supplying the name of the place-holder field with the POSITION attribute.

The syntax for an OCCURS segment is

```
SEGNAME=occseg, PARENT=imsseg, OCCURS=occurs, $
```

where:

occseg Is the name of the OCCURS segment. It can be any valid segment name.

imsseg Is the name of the base segment. It must be the IMS segment name.

occurs Indicates the number of repetitions.. Possible values are as follows:

n is the fixed number of times that the group repeats in the segment. It is an integer value from 1 to 4095.

n, *POSITION=posfield* signals that the repeating field is embedded within the base segment rather than occurring at the end and names a field in the base segment that marks the starting position of the repeating field.

nfield is the name of a field in the parent or non-repeating portion of the segment whose value is the number of times that the group repeats. You must define this field in the Master File whether or not it is a search field defined in the DBD.

VARIABLE indicates that the length of the repeating segment varies and that the number of occurrences can be computed from each segment. In this case, the (base) segment must contain a counter field as its first field; the counter field's alias value must be IMSname.CNT (IMSname is the name of the field in the IMS DBD).

In the IMS DBD, a variable length segment differs from a fixed length segment only in the BYTES parameter. For variable length segments, the BYTES parameter consists of two values: the maximum and minimum number of bytes. IMS cannot search for values among the repetitions within a segment. Therefore, in any FOCUS retrieval request that references a field in a repeating group, FOCUS, and not IMS, searches and screens the OCCURS segments.

The ORDER Field

Sometimes the sequence of fields within an OCCURS segment is significant. For example, each instance of the repeating field may represent one quarter of the year, but the segment may not have a field that identifies to which quarter it applies.

ORDER is an optional counter used to identify the sequence number within a group of repeating fields. You should include it when the order of data is important. The ORDER field does not represent an existing field in the database; it is used only for internal processing.

The ORDER field must be the last field described in the OCCURS segment. The syntax is

```
FIELDNAME=fieldname, ALIAS=ORDER, USAGE=In, ACTUAL=I4 , $
```

where:

fieldname Is a valid fieldname.

In Is an integer format.

Note:

- The ALIAS value must be ORDER.
- The ACTUAL format must be I4.
- The ORDER field must be the last field defined in the OCCURS segment.

In report requests, you can use the value of the ORDER field. You can also use a DEFINE field in your session or in the Master File to translate it to more meaningful values. For example:

```
DEFINE ...  
QTR/A3 = DECODE ORDER(1 '1ST' 2 '2ND' 3 '3RD' 4 '4TH')  
.  
.  
.
```

A subsequent request could include:

```
PRINT TOT.TAXES IF QTR IS 1ST
```

or

```
PRINT QTR BALANCE INTEREST
```

Example: OCCURS=n

In the following example, the IMS segment, IMS1, includes a group (consisting of the two fields MONTH and AMOUNT) that repeats 12 times. The COBOL FD for the segment is shown:

```
01  IMS1
   05  ACCOUNT  PIC X(9)
   05  TYPE     PIC XXX
   05  PAYMENT  OCCURS 12 TIMES
       10 MONTH PIC 99
       10 AMT   PIC S9(3)V(99)COMP-3
```

The Master File uses two segments to describe this IMS variable length segment:

1. SEGNAME=IMS1 , PARENT= , SEGTYPE=S2
 FIELD=ACT_NUM , ALIAS=ACCOUNT.KEY , A9 , A9 , \$
 FIELD=TYPE , ALIAS=TYPE , A3 , A3 , \$
2. SEGNAME=OCC1 , PARENT=IMS1 , OCCURS=12
 FIELD=MM , ALIAS=MONTH , A2 , A2 , \$
 FIELD=AMT , ALIAS=AMT , P6.3 , P3 , \$

1. Segment IMS1 is the base segment. It has the same name as the IMS segment and describes the two non-repeating fields: ACT_NUM and TYPE.
2. The OCCURS segment, OCC1, identifies IMS1 as its parent. It has no SEGTYPE, and it includes the OCCURS attribute. The two repeating fields are described in this segment.

In the following example, the repeating group is not at the end of the segment, but rather is embedded in the segment before the LNAME field. The COBOL FD for this situation is shown:

```
01  IMS1
   05  ACCOUNT  PIC X(9)
   05  TYPE     PIC XXX
   05  PAYMENT  OCCURS 12 TIMES
       10 MONTH PIC 99
       10 AMT   PIC S9(3)V(99)COMP-3
   05  LNAME    PIC X(20)
```

Creating FOCUS Descriptions

The Master File must include LNAME in the base segment. It must also describe where the repeating fields fit into the base segment by defining a place-holder field before LNAME, equal to the length of the 12 occurrences and by pointing to the place-holder field with the POSITION attribute:

```
SEGNAME=IMS1 , PARENT= , SEGTYPE=S2
FIELD=ACT_NUM , ALIAS=ACCOUNT.KEY , A9 , A9 , $
FIELD=TYPE , ALIAS=TYPE , A3 , A3 , $
1. FIELD=HOLDIT , ALIAS= , A60 , A60 , $
FIELD=LNAME , ALIAS=LAST_NAME , A20 , A20 , $
2. SEGNAME=OCC1 , PARENT=IMS1 , OCCURS=12 , POSITION=HOLDIT
FIELD=MM , ALIAS=MONTH , A2 , A2 , $
FIELD=AMT , ALIAS=AMT , P6.3 , P3 , $
```

1. The HOLDIT field is the place holder for the repeating group in the base segment (IMS1). Since the repeating group consists of 12 occurrences, each of which is 5 bytes long (A2 and P3, described in segment OCC1), the HOLDIT field is defined as A60.
2. The attribute POSITION=HOLDIT in the OCCURS segment declaration describes where the repeating group is located in the actual (base) segment.

Example: OCCURS=fieldname

In the next example, the value of the TIMES field is the number of occurrences. The following COBOL FD describes this situation:

```
01 IMS1
05 ACCOUNT PIC X(9)
05 TYPE PIC XXX
05 TIMES PIC S999 COMP-3
05 PAYMENT OCCURS DEPENDING ON TIMES
10 MONTH PIC 99
10 AMT PIC S9(3)V(99)COMP-3
```

The Master File attribute, OCCURS=TIMES, identifies the TIMES field as containing the number of repetitions. The Master File also defines the optional ORDER field as the last field in the OCCURS segment:

```
SEGNAME=IMS1 , SEGTYPE=S2
FIELD=ACT_NUM , ALIAS=ACCOUNT.KEY , A9 , A9 , $
FIELD=TYPE , ALIAS=TYPE , A3 , A3 , $
FIELD=TIMES , ALIAS=TIMES , P4 , P2 , $
1. SEGNAME=OCC1 , PARENT=IMS1 , OCCURS=TIMES
FIELD=MM , ALIAS=MONTH , A2 , A2 , $
FIELD=AMT , ALIAS=AMT , P6.3 , P3 , $
2. FIELD=WHICH , ALIAS=ORDER , I4 , I4 , $
```

1. The attribute OCCURS=TIMES identifies the value in the TIMES field as the number of instances of the repeating group.
2. The field named WHICH is the optional ORDER field (ALIAS=ORDER). It is an internal counter defined as the *last* field in the OCCURS segment. It associates a sequence number with each occurrence of the repeating group. With the ORDER field defined, a request can include a test that selects a specific occurrence; for example:

```
WHERE WHICH EQ 3
```

Example: OCCURS=VARIABLE

This example describes a segment in which the number of occurrences must be calculated from the length of the segment. The first field in the segment must be a 2-byte counter field that contains the true length of the segment and is defined to IMS in the DBD. The COBOL FD for this variable length segment is as follows:

```
01  IMS1
   05  COUNTER PIC 99 COMP
   05  ACCOUNT PIC X(9)
   05  TYPE    PIC XXX
   05  PAYMSCHED OCCURS 1 TO 12 TIMES
       10 MONTH PIC 99
       10 AMT   PIC S9(3)V(99)COMP-3
```

The Master File must specify OCCURS=VARIABLE and must describe the counter field with the attribute ALIAS=IMSname.CNT:

```
SEGNAME=IMS1 , SEGTYPE=S2
1.  FIELD=COUNTFLD , ALIAS=COUNTER.CNT , I2 , I2 , $
    FIELD=ACT_NUM , ALIAS=ACCOUNT.KEY , A9 , A9 , $
    FIELD=TYPE , ALIAS=TYPE , A3 , A3 , $
2.  SEGNAME=OCC1 , PARENT=IMS1 , OCCURS=VARIABLE
    FIELD=MM , ALIAS=MONTH , A2 , A2 , $
    FIELD=AMT , ALIAS=AMT , P6.3 , P3 , $
```

1. The 2-byte counter field must be the first field in the base segment. Its alias is formed by appending the suffix CNT to the field name defined in the IMS DBD: ALIAS=COUNTER.CNT.
2. In the OCCURS segment definition, the attribute OCCURS=VARIABLE indicates that the first field defined in the Master File contains the segment length and that the number of repetitions must be calculated from this length.

3.3 The Access File

In most environments, you select a PSB prior to invoking FOCUS. However, if you connect to IMS using the DBCTL environment, you select the PSB during your session, and you have the option of switching PSBs between requests.

You have two options for selecting a PSB within the session:

- You can issue the IMS SET PSB command either from the command line, within your FOCUS or MSO profile, or in a FOCEXEC (see the instructions for implementing DBCTL in Chapter 5, *Environments*).
- You can use an Access File to identify the PSB. Each Access File corresponds to one Master File; it identifies the PSB associated with its corresponding Master File.

If you use an Access File, the Interface identifies the appropriate PSB when you reference a Master File in a request. The selection is automatic and transparent to you.

Each Access File contains one attribute starting within the first eight bytes. The syntax is

```
PSB=psbname, $
```

where:

psbname Indicates the name of the FOCPSB library member to use. This name must be identical to the name of the actual PSB that IMS will access. If the member does not exist, the following error message is generated:

```
(FOC4261) FOCPSB MEMBER NOT FOUND: psbname
```

Access Files are members of a partitioned dataset. The member name of an Access File within its partitioned dataset must be the same as the member name of the corresponding Master File within *its* partitioned dataset. At run time, the Access File dataset is allocated to ddname ACCESS. (This allocation is optional if all PSB selection is dynamic.)

Note:

- All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following error message:

```
(FOC4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```

- The IMS SET PSB command supersedes the PSB attribute in the Access File.

The following is a sample Access File corresponding to the DI21PART Master File. The Access File is member DI21PART in the Access File library:

```
PSB=FOCSD, $
```

4 Reporting Efficiencies

This chapter explains how the Interface optimizes report generation. It includes the following topics:

- Interface optimization (see Section 4.1, *Interface Optimization*).
- How the Interface constructs DL/I calls (see Section 4.2, *DL/I Calls*).
- Record selection tests (see Section 4.3, *Record Selection Tests*).
- Joins (see Section 4.4, *The Dynamic JOIN Command*).
- Retrieval of unique segments (see Section 4.5, *Retrieval of Unique Segments*).
- JOINS with selection criteria (see Section 4.6, *JOINS With Selection Criteria*).

Note:

- See the *FOCUS for IBM Mainframe Users Manual* for information about retrieval of parent segments that lack descendant instances (short paths) and retrieval of segments from multiple paths in the hierarchy (multi-path requests).
- When you issue a request in a shared database environment, if an IMS call returns a GG status code (indicating an enqueue), the Interface attempts to retrieve the segment again. The request does not fail unless the Interface detects the GG status code ten times in succession.

If the call still fails after 10 attempts, the user receives a message indicating that the report is incomplete, and the report is displayed with all records retrieved up until that time.

This chapter includes sample report requests. Refer to Appendix A, *Sample File Descriptions*, for complete file descriptions of the databases referenced in these examples.

4.1 Interface Optimization

When you issue a FOCUS report request, the Interface must determine how much of the request it can pass off to IMS (that is, translate into an IMS SSA). If IMS can apply the screening conditions and select the proper records from the database, FOCUS need only format the results. However, if the Interface cannot translate parts of the FOCUS request into an SSA, it must retrieve the segments sequentially from IMS and let FOCUS apply the screening criteria.

Optimization is the process by which the Interface passes the selection operations of a FOCUS request to IMS for processing. Interface optimization reduces the volume of IMS-to-FOCUS communication and improves response time by exploiting IMS internal optimization techniques.

A qualified SSA (Segment Search Argument) for a segment is a logical condition on one or more sequence or search fields from the segment. IMS retrieves only the segment instances that make the condition true. For example, a qualified SSA on the STANINFO segment of the DI21PART database could be the following:

```
STANINFO(STANKEY EQ AA)
```

The Interface communicates with IMS by issuing GU (Get Unique) and GN (Get Next) DL/I calls. (See Chapter 2, *IMS Overview and Mapping Concepts*, if you are unfamiliar with the terms and concepts mentioned in this chapter.) The most efficient DL/I calls incorporate qualified SSAs and thus enable IMS to find the required segment instances. Therefore, when the Interface intercepts a FOCUS report request, its goal is to construct a set of qualified SSAs that completely describes the FOCUS screening criteria and to pass these SSAs to IMS.

Although you can issue any FOCUS report request through the Interface, keep the following in mind:

- Only fields that are IMS sequence fields, search fields, or secondary indexes (in the Master File, ALIAS=IMSname.KEY, .IMS, .HKY, or .SKY) can generate qualified SSAs. If the FOCUS request includes selection criteria on fields that are not sequence or search fields, the Interface retrieves the necessary segments sequentially, and FOCUS screens them.
- Screening conditions on fields in OCCURS segments cannot be optimized.
- Screening conditions on group fields can be optimized only if your request references either the group name or the left-most component fields (high-order fields) from the group. Secondary index fields are group fields and are subject to this same restriction.

See Section 4.3, *Record Selection Tests*, for a detailed discussion of screening conditions and optimization.

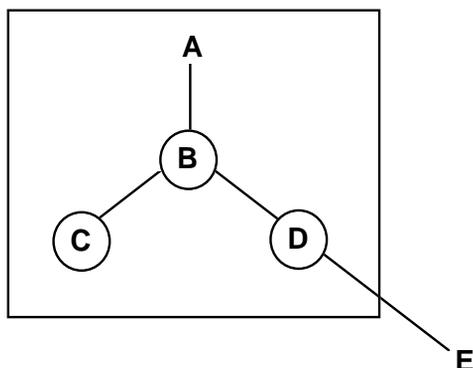
4.2 DL/I Calls

The Interface operates in two phases:

- For each report request, the setup phase analyzes the query and builds arguments for DL/I calls.
- The execution phase issues the DL/I calls using appropriate arguments based on the current position in the database and feedback from FOCUS regarding the last retrieved segment.

The Interface retrieves data from the IMS database with a sequence of GET UNIQUE and GET NEXT calls that follow the database structure in top-to-bottom, left-to-right order. These calls do not require a PCB that maintains multiple positions in the database. The Interface does not issue path calls (calls that retrieve a hierarchical path, not just one segment) because those might retrieve unwanted segments, burdening IMS with unnecessary I/O. Therefore, the Interface can use any PCB.

The DL/I calls that the Interface issues retrieve the smallest connected subtree, starting at the root, that contains all explicitly referenced fields plus the root. For example:



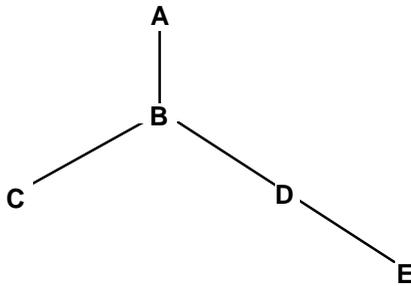
If the request explicitly references only fields in segments B, C and D, the Interface retrieves all the segments in the boxed area, top-to-bottom, left-to-right.

Descendants of rejected segments do not participate in the retrieval. When FOCUS applies screening conditions that were not translated into SSAs, it communicates to the Interface the status of the previously retrieved segment: accepted or rejected. If the segment was rejected, the next DL/I call is for either another instance of the rejected segment (TWIN), a sibling segment, or a parent segment, not a child of the rejected segment.

Reporting Efficiencies

The Interface accesses a segment, even when the request does not reference it, if the request references any of its descendants. For example:

```
TABLE FILE PEOPLE
COUNT C AND E
IF B CONTAINS 'ABC'
IF EKEY IS 1 OR 2
END
```



The request references fields in segments B and E, so the Interface also retrieves segment D. It retrieves segment A for efficient positioning. It cannot translate the condition on B to an SSA; it translates the condition on the key field, E, to two SSAs: (EKEY EQ 1) and (EKEY EQ 2).

4.3 Record Selection Tests

You can issue any FOCUS report request through the IMS/DB Interface. However, those requests whose record selection criteria can be applied at the IMS level (with qualified SSAs on .KEY, .IMS, .HKY, or .SKY fields) result in many fewer DL/I calls and I/O operations than non-optimized requests. They can achieve performance improvements measured in orders of magnitude.

Note: Since IMS does not support the concept of missing data, FOCUS considers all fields not missing.

This section includes sample requests with their corresponding FSTRACE4 results. When you allocate FSTRACE4 as described in Appendix B, *Tracing Interface Processing*, you get a dump of the SSA buffer set-up for each request. For example, the following is a selection test on the SEQFIELD search field in the PATINFO segment of the PATDB01 database (see Appendix A, *Sample File Descriptions*):

```
IF SEQFIELD EQ '100000' OR '100001'
```

It produces the following FSTRACE4 results:

```

set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 C5D8C6C9      *PATINFO *-(SEQFI*
C5D3C47E 40F1F0F0 F0F0F04E E2C5D8C6      *ELD= 100000+SEQF*
C9C5D3C4 7E40F1F0 F0F0F0F1 5D40          *IELD= 100001)  *

```

Note:

- The dump has a left side and a right side. The left side shows the hexadecimal values in the memory locations. The right side, delimited by *'s, contains the corresponding alphanumeric characters.

Since not all hexadecimal values represent printable alphanumeric characters, those parts of the dump that correspond to non-printable characters display as blanks on the right side. In particular, if a comparison value in an SSA is a packed number, it displays as blank on the right-hand side of the dump. In this example, the comparison values are alphanumeric and, therefore, they display.

- The logical operator AND is represented by an asterisk (*) in the SSA, and the logical operator OR is represented by a plus sign (+).
- The field names in the SSA are the IMS names for the fields; the Interface finds these names in the Master File as ALIAS values.
- The SSA always includes the segment name, PATINFO in this case. What distinguishes an optimized request from a non-optimized request is whether the SSA is qualified; a qualified SSA includes the selection test from the request, as in the previous example. An unqualified SSA contains the segment name with no selection criteria.
- If you access the Interface in the DBCTL environment, the trace includes additional statistics. Appendix B, *Tracing Interface Processing*, contains a detailed description of trace facilities.

Whenever possible, the Interface translates screening conditions from the FOCUS request into qualified SSAs. However, not all FOCUS screening conditions can be translated into SSAs. Such conditions are still applied, but by FOCUS, not by IMS. In those cases, the Interface retrieves the data sequentially and passes it to FOCUS for screening.

4.3.1 Access Method Restrictions

Some IMS access methods are limited in their ability to apply certain screening conditions. These restrictions can affect request optimization. The following considerations apply to specific access methods:

- Since HDAM databases are stored randomly and accessed directly, IMS cannot retrieve their records in root key sequence. Therefore, with an HDAM database, the Interface can only optimize equality conditions on the root key:

`KEY EQ value`

You can build a secondary index on the root key and use it to optimize sequential (range) processing. Section 4.3.6, *Auto Index Selection*, contains an example.

Note: The NE condition is not optimized.

- With HIDAM databases, the Interface optimizes LIKE tests with a mask by translating the LIKE into a range condition. See Section 4.3.5, *Partial Key and Multi-Segment Requests*, for an example.
- To take advantage of efficiencies available by limiting processing to one area of a Fast Path DEDB database, the PSB must limit access to that area.

4.3.2 Rules for Constructing SSAs From FOCUS IF Tests

In order for the Interface to translate a FOCUS IF test into a qualified SSA, the following conditions must exist:

- The field tested must be an IMS key field, search field, or a secondary index.
- The test relation must be one of the relations described in this section.
- The tested field cannot be a floating point numeric or zoned field.

Note: In order for the Interface to optimize selection tests on zoned key fields, you must describe the USAGE format of these fields as alphanumeric or packed in the Master File.

- The ACTUAL and USAGE formats of the tested field must be basically the same, so no format conversion is required. For example, extending with leading zeros or padding with trailing blanks is not a format conversion, but shifting the implied decimal point in a packed field is.
- In an HDAM database, record selection tests must use the EQ relation on the root key, with no value masking, or you must have a secondary index on the root key.

The Interface can translate the following test relations to qualified SSAs:

- Comparison of a field to a list of values:

field relation value1 OR value2 ... OR valuen

See Section 4.3.3, *Complex Screening Conditions*, for a discussion of SSA generation when comparing a field to a list of values.

- IS, EQ
- IS-NOT, NE
- IS-FROM, FROM, GE
- LIKE
- NOT LIKE
- TO, LE
- FROM...TO

The Interface treats the FROM...TO relation as a pair of relations. For example:

IF field GE . . . AND IF field LE . . .

The Interface also translates a search on a partial key (the high-order portion of a key) to a range condition and optimizes it (see Section 4.3.5, *Partial Key and Multi-Segment Requests*, for an example).

- NOT-FROM...TO
- IS-MORE-THAN, EXCEEDS, GT
- IS-LESS-THAN, LT

Note: The Interface does not optimize the relations INCLUDES, EXCLUDES, CONTAINS, and OMTS.

4.3.3 Complex Screening Conditions

The general form for a complex screening condition is:

field1 relation1 value1 OR field2 relation2 value2 OR ...

If the fields are search or sequence fields, the Interface can optimize the screening test subject to certain conditions. The Interface either constructs a single SSA or multiple SSAs, depending on the characteristics of the segment and the type of relation used.

Note: Using the AND operator between logical conditions in a selection test is equivalent to using multiple IF statements without the AND operator. The Interface constructs a qualified SSA that incorporates the AND operation in either case.

This section describes the following:

- The SSA buffer, and how the Interface processes SSAs that do not fit into the buffer.
- Conditions under which the Interface constructs a single SSA.
- Conditions under which the Interface constructs multiple SSAs.

The SSA Buffer

Once the Interface constructs an SSA, it must place it in the SSA buffer in order to submit it in a DL/I call. If the SSA is too long to fit into the buffer, the Interface makes the following choices between the individual screening conditions within the SSA:

- Tests on key fields take precedence over all other tests.
- The Interface gives precedence to remaining tests based on their order in the query. It retains earlier tests and eliminates later tests from the SSA, so you can influence the conditions included in the SSA by carefully constructing your screening conditions.

If an SSA does not fit into the SSA buffer, no error is generated. The selection tests omitted from the SSA are applied, but they are not optimized; that is, FOCUS applies them, not IMS.

Note: The number of conditions that fit into the SSA buffer is not fixed; it varies depending on the lengths of the values in the comparisons and on the relations used in the comparisons.

Constructing a Single SSA

The Interface constructs a single SSA that incorporates the entire complex logical condition if any of the following is true:

- The segment has no key.
- There is a key, but the request does not include a screening condition that references it.
- There is a record selection test on the key that specifies the EQ relation or a single range.

If the entire SSA fits into the SSA buffer, the Interface submits it in the DL/I call. If the SSA is too long to fit into the SSA buffer, the Interface makes the choices described in the previous section, *The SSA Buffer*.

The following example illustrates SSA generation when there is no equality test on the key field, but there are tests on a search field:

```
TABLE FILE PATINFO
PRINT SSN SEQFIELD LAST_NAME
IF SEQFIELD EQ '100000' OR '100005'
IF LAST_NAME EQ 'BORRERO' OR 'JONES'
END
```

These selection criteria produce the following trace:

set up SSA-Q:	
D7C1E3C9 D5C6D640 5C604DE2 C5D8C6C9	*PATINFO *-(SEQFI*
C5D3C47E 40F1F0F0 F0F0F05C D3D5C1D4	*ELD= 100000*LNAM*
C5404040 7E40C2D6 D9D9C5D9 D6404040	*E = BORRERO *
40404EE2 C5D8C6C9 C5D3C47E 40F1F0F0	* +SEQFIELD= 100*
F0F0F05C D3D5C1D4 C5404040 7E40D1D6	*000*LNAME = JO*
D5C5E240 40404040 40404EE2 C5D8C6C9	*NES +SEQFI*
C5D3C47E 40F1F0F0 F0F0F55C D3D5C1D4	*ELD= 100005*LNAM*
C5404040 7E40C2D6 D9D9C5D9 D6404040	*E = BORRERO *
40404EE2 C5D8C6C9 C5D3C47E 40F1F0F0	* +SEQFIELD= 100*
F0F0F55C D3D5C1D4 C5404040 7E40D1D6	*005*LNAME = JO*
D5C5E240 40404040 40405D40	*NES) *

The trace shows that the Interface generates one SSA, incorporating the following selection criteria:

```
(SEQFIELD EQ 100000 AND LNAME EQ BORRERO
OR SEQFIELD EQ 100000 AND LNAME EQ JONES
OR SEQFIELD EQ 100005 AND LNAME EQ BORRERO
OR SEQFIELD EQ 100005 AND LNAME EQ JONES)
```

If this SSA did not fit into the SSA buffer, the Interface would retain as much of it as possible in a qualified call, after which FOCUS would apply the remaining tests to the returned segments.

Reporting Efficiencies

The request produces the following report:

PAGE	1		
SSN	SEQFIELD	LAST_NAME	
---	-----	-----	
197548684	100005	JONES	

Constructing Multiple SSAs

This section describes how the Interface handles SSA generation when the condition in the FOCUS request compares a key field to a list of values. The key can be unique or non-unique. The form of such a condition is

```
IF key EQ value1 OR value2 OR value3 ...
```

```
IF key IS (filename)
```

or

```
WHERE key IN (value1, value2, ... ,valuen)
```

where:

key Is a key field or secondary index.

value1...valuen Are the comparison values.

filename Indicates that the comparison values are stored in a sequential file allocated to DDNAME *filename*. The file can contain up to 4000 bytes of data; for an 8-byte key, this is enough space for 500 values.

With this kind of screening condition, the Interface constructs a separate SSA for each value in the list, in ascending sort sequence, and transmits each one in turn to IMS:

```
(key EQ value1)  
.  
.  
.  
(key EQ valuen)
```

The Interface first issues a DL/I call containing only the first SSA. If IMS locates a segment that satisfies the condition in the SSA, the Interface returns the segment to FOCUS. Otherwise, the Interface issues a DL/I call that incorporates only the second SSA. It continues until IMS either locates a segment that satisfies one of the SSAs or exhausts the list of values.

In the following example, the Interface constructs three SSAs:

```
TABLE FILE PATINFO
PRINT SSN SEQFIELD LAST_NAME
IF SSN EQ '197548682' OR '197548685' OR '197548691'
END
```

The FSTRACE4 results show the three separate SSAs generated:

```
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F8F25D40 * = 197548682) *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F8F55D40 * = 197548685) *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F9F15D40 * = 197548691) *
```

The following is the resulting report:

```
PAGE      1

SSN          SEQFIELD  LAST_NAME
---          -
197548682    100003    SALEH
197548685    100006    JACA
197548691    100012    BOYCE
```

The next request illustrates an equality test on the key field and an additional test on a search field:

```
TABLE FILE PATINFO
PRINT SSN SEQFIELD LAST_NAME
IF SSN EQ '197548679' OR '197548682' OR '197548685'
IF SEQFIELD GT '100000'
END
```

Reporting Efficiencies

The request produces multiple SSAs, as illustrated in the following trace:

```
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F7F95CE2 * = 197548679*S*
C5D8C6C9 C5D3C46E 40F1F0F0 F0F0F05D *EQFIELD> 100000)*
40 * *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F8F25CE2 * = 197548682*S*
C5D8C6C9 C5D3C46E 40F1F0F0 F0F0F05D *EQFIELD> 100000)*
40 * *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F8F55CE2 * = 197548685*S*
C5D8C6C9 C5D3C46E 40F1F0F0 F0F0F05D *EQFIELD> 100000)*
40 * *
```

The Interface constructs three SSAs and applies them one at a time:

```
(SSN EQ 197548679 AND SEQFIELD GT 100000)
(SSN EQ 197548682 AND SEQFIELD GT 100000)
(SSN EQ 197548685 AND SEQFIELD GT 100000)
```

The following report is produced:

PAGE	1	
SSN	SEQFIELD	LAST_NAME
---	-----	-----
197548682	100003	SALEH
197548685	100006	JACA

If the SSA generated by combining the conditions in all the IF statements is too long to fit into the SSA buffer, the Interface retains as much of it as possible in a qualified call by applying the rules described in the previous section, *The SSA Buffer*.

Sequentially Accessed Root Segments

If the root segment is the target of the SSA generated by a request, and if it has a unique key, the Interface assumes that IMS will use an index or randomizing scheme to locate the segment without an exhaustive search of the root segment chain. Therefore, the Interface retrieves the segment with qualified GET UNIQUE calls.

Even if the assumption that there is an index or randomizing scheme for IMS to use is not valid, as with HSAM databases, each call starts its search at the first record in the database. In this situation, it is preferable to describe the root segment as having no key (SEGTYPE=S0), and not to describe any field's alias with the .KEY suffix. This technique causes the Interface to issue qualified GET NEXT calls that access the roots sequentially and maintain their position in the database from one call to the next.

4.3.4 WHERE Tests

When the Interface constructs DL/I calls, it looks only at IF selection tests, not WHERE tests. However, prior to constructing DL/I calls, it examines WHERE selection tests and, when possible, translates them to equivalent IF statements. Therefore, WHERE tests that can be expressed as IF tests are subject to the same optimization rules as IF tests.

For example, if field1 and field2 are search or sequence fields, the Interface optimizes the following WHERE test:

```
WHERE field1 relation1 value1 AND field2 relation2 value2
```

It is equivalent to the following:

```
IF field1 relation1 value1
IF field2 relation2 value2
```

The Interface can also optimize the following WHERE conditions:

- WHERE tests that use the AND operator between adjacent parts of a key.
- WHERE tests that compare an IMS field to a list of values, as described in *Constructing Multiple SSAs* in Section 4.3.3, *Complex Screening Conditions*.

The Interface optimizes the following request:

```
TABLE FILE PATINFO
PRINT SSN SEQFIELD LAST_NAME
WHERE SSN EQ '197548679' AND SEQFIELD EQ '100000'
END
```

Reporting Efficiencies

The trace shows the qualified SSA that the request generates:

```
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040 *PATINFO *-(SSN *
4040407E 40F1F9F7 F5F4F8F6 F7F95CE2 * = 197548679*S*
C5D8C6C9 C5D3C47E 40F1F0F0 F0F0F05D *EQFIELD= 100000)*
40 * *
```

The report follows:

```
PAGE      1

SSN        SEQFIELD  LAST_NAME
---        -
197548679  100000      ROSANO
```

The Interface cannot optimize the following types of WHERE tests:

```
WHERE field1 relation field2
WHERE field1 relation1 value1 OR field2 relation2 value2
```

For example, the Interface does not optimize the next request:

```
TABLE FILE PATINFO
PRINT SSN SEQFIELD LAST_NAME
WHERE SSN EQ '197548679' OR SEQFIELD EQ '100000'
END
```

The resulting SSA is not qualified; it includes the segment name, but no selection criteria:

```
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604040 *PATINFO *- *
```

FOCUS retrieves all records from the segment, applies the selection criteria, and produces the following report:

```
PAGE      1

SSN        SEQFIELD  LAST_NAME
---        -
197548679  100000      ROSANO
```

4.3.5 Partial Key and Multi-Segment Requests

This section illustrates SSAs for requests that select on a partial key and requests that access values from multiple segments.

Selection on a Partial Key

The Interface can optimize record selection on a partial key unless the database is an HDAM database. The partial key must be the high-order (leftmost) portion of the key. To search on a partial key, use a mask as the comparison value in the relation. The syntax is

```
{IF|WHERE} field EQ 'xxx$*'
{IF|WHERE} field LIKE 'xxx%'
```

where:

<i>field</i>	Is a valid field name.
<i>xxx</i>	Are any number of characters that constitute the leftmost portion of the key. You must enclose the masking characters between single quotation marks.
<i>\$* or %</i>	Are wildcard characters indicating that any string of characters in this position satisfies the screening criteria. Use \$* with the EQ relation and % with the LIKE relation.

The Interface translates the condition to a range using GE and LE. Consider the following request:

```
DYNAM ALLOC FILE FSTRACE4 DA *
TABLE FILE PATINFO
PRINT SSN SEQFIELD LAST_NAME
WHERE SSN EQ '1975486$*'
END
```

The FSTRACE4 results show that the Interface translated the selection test to a range condition:

```
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040      *PATINFO *-(SSN *
4040406E 7EF1F9F7 F5F4F8F6 00005CE2      *   >=1975486 *S*
E2D54040 4040404C 7EF1F9F7 F5F4F8F6      *SN    <=1975486*
FFFF5D40                                     *::)      *
```

To define the range of values, the Interface appends the lowest hexadecimal value (00) and the highest hexadecimal value (FF) to 1975486 (00 and FF are not alphanumeric representations of numbers; therefore they do not print on the right side of the dump).

Reporting Efficiencies

The first few lines of the resulting report follow:

PAGE	1		
SSN	SEQFIELD	LAST_NAME	
---	-----	-----	
197548679	100000	ROSANO	
197548681	100002	BORRERO	
197548682	100003	SALEH	
197548683	100004	SALGADO	
197548684	100005	JONES	
197548685	100006	JACA	
197548686	100007	PENA	
197548687	100008	FREEMAN	
.			
.			
.			

Multi-Segment Requests

When a request requires access to multiple segments, the Interface constructs one SSA for each segment. (Section 4.2, *DLI Calls*, explains the rules for accessing multiple segments.) If the selection criteria for a particular segment can be optimized, its corresponding SSA is qualified.

The following request includes an equality test on PARTKEY, the root key of the DI21PART database, but it contains no selection criteria for the STANKEY field in the STANINFO segment:

```
TABLE FILE DI21PART
PRINT PARTKEY STANKEY
IF PARTKEY EQ '02AN960C10'
END
```

The SSA for the root segment, PARTROOT, is qualified, but the SSA for the STANINFO segment is not qualified:

set up SSA-Q:			
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2	*PARTROOT*-(PARTK*		
C5E8407E 40F0F2C1 D5F9F6F0 C3F1F040	*EY = 02AN960C10 *		
40404040 40405D40	*) *		
set up SSA-Q:			
E2E3C1D5 C9D5C6D6 5C604040	*STANINFO*-	*	

The request produces the following report:

PAGE	1
PARTKEY	STANKEY
-----	-----
02AN960C10	02

The next request includes a not-equal condition on PARTKEY, the root key of the DI21PART database; it has no selection criteria for STANKEY:

```
TABLE FILE DI21PART
PRINT PARTKEY STANKEY
IF PARTKEY NE '02AN960C10'
END
```

The following is the SSA for both the PARTROOT and STANINFO segments:

```
set up SSA-Q:
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2 *PARTROOT*-(PARTK*
C5E8405F 7EF0F2C1 D5F9F6F0 C3F1F040 *EY ^=02AN960C10 *
40404040 40405D40 * ) *
set up SSA-Q:
E2E3C1D5 C9D5C6D6 5C604040 *STANINFO*- *
```

FOCUS applies the selection criteria to produce the desired report. The first several lines in the report follow:

PAGE	1
PARTKEY	STANKEY
-----	-----
02CK05CW181K	02
02CSR13G104KL	02
02JAN1N976B	02
02MS16995-28	02
02N51P3003F000	02
02RC07GF273J	02
02TPART01	02
02106B1293P009	02
02250236-001	02
.	
.	
.	

Reporting Efficiencies

It is important to consider how qualified SSAs on lower level segments of a database may affect performance:

- IMS retrieves lower level segments sequentially (after possibly locating the appropriate root segment via an index or hash code).
- If the Interface passes a qualified SSA on a lower level segment to IMS without a qualified SSA on the root segment, IMS does not return data or a return code to the Interface until it either finds an appropriate segment or exhausts the entire database.
- Depending on the size of the database, the Interface could wait an extremely long time between calls without any way to interrupt the process.

These facts make it advantageous to screen on the root segment, where an index or hash code makes retrieval efficient. One technique for preventing an exhaustive search of the database is to define fields in low level segments in the Master File as non-search fields. This approach forces control to return to FOCUS after retrieval of each segment instance, giving you the opportunity to limit processing with a READLIMIT test. For information on READLIMIT, see Section 4.3.7, *Search Limits*.

4.3.6 Auto Index Selection

When the Master File and FOCPSB define secondary indexes for a database, the Interface analyzes each request to determine the most efficient entry point into the database. For information on creating a Master File and FOCPSB for use with a database that has secondary indexes, refer to Chapter 2, *IMS Overview and Mapping Concepts*, and Chapter 3, *Creating FOCUS Descriptions*.

The Interface scans each request to determine if fields used in record selection tests are key fields, secondary indexes, or the high-order (leftmost) parts of either. Depending on the request criteria, the Interface selects the appropriate PCB for the most efficient access to the data. The PATINFO Master File describes a secondary index named IXADMD on the ADMIT_DATE field.

In the following request, the field referenced in the IF condition is the field associated with the secondary index called IXADMD:

```
TABLE FILE PATINFO
PRINT LAST_NAME SALARY ADMIT_DATE
IF ADMIT_DATE EQ '19920925'
END
```

The trace shows that the Interface generates a qualified SSA using the IXADMD index:

```
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DC9 E7C1C4D4      *PATINFO *-(IXADM*
C440407E 40F1F9F9 F2F0F9F2 F55D40      *D = 19920925) *
```


Reporting Efficiencies

Now, the Interface can use the Auto Index Selection feature on the same request to generate a qualified SSA:

```
> > TABLE FILE AIHDAM
> PRINT *
> WHERE EMPLOYEE_ID6 FROM 5248 TO 6393
> END
```

```
set up SSA-Q:
D3C1D5C7 E4C1C7C5 5C604DC9 E7C5D4D7 *LANGUAGE*-(IXEMP*
F640406E 7E000014 805CC9E7 C5D4D7F6 *6 >= Ø*IXEMP6*
40404C7E 000018F9 5D40 * <= 9) *
```

4.3.7 Search Limits

There are two search limit tests available through the Interface.

- READLIMIT limits the number of records searched in a file.
- RECORDLIMIT stops a search after finding a specified number of records that pass all screening criteria.

When reporting against an IMS database, you may want to halt the retrieval after reading part of the database when any of the following conditions exists:

- You are testing a new Master File. A few records are adequate to determine whether the description is correct.
- You are testing a new report format.
- You need to generate a trace for debugging purposes.
- The Database Administrator wants to limit segment search and retrieval for very large IMS databases.

The syntax is

```
IF {READLIMIT|RECORDLIMIT} EQ n
```

where:

n Is a number greater than 0.

The Database Administrator can supply this screening condition directly in the Master File, in which case you cannot override it. If you include a READLIMIT or RECORDLIMIT test in a request, and the Database Administrator also includes one in the Master File, FOCUS uses the smaller value. (Refer to the *FOCUS for IBM Mainframe Users Manual* for additional information on READLIMIT and RECORDLIMIT.)

FOCUS, not IMS, applies READLIMIT and RECORDLIMIT tests. Therefore, because IMS always searches as much of the database as necessary to locate an appropriate segment, the READLIMIT setting has no effect on retrieval in an optimized request.

To illustrate this point, the next examples reference fields from the PATINFO Master File (see Appendix A, *Sample File Descriptions*). The database contains the following values:

SEQFIELD	FIRST_NAME	LAST_NAME
100000	ANDRE	ROSANO
100001	LARRY	LOVELACE
100002	JOHN	BORRERO
100003	JAMAL	SALEH

Consider this request:

```
TABLE FILE PATINFO
PRINT FNAME
IF READLIMIT EQ 1
END
```

Since the field FNAME is not qualified, FOCUS applies the READLIMIT condition and locates the following value for FNAME in its one read:

```
ANDRE
```

The next request includes a screening condition on FNAME, an IMS search field. The Interface passes the screening condition to IMS in a qualified SSA, and IMS searches for a segment that passes the test, unaware of the READLIMIT condition:

```
TABLE FILE PATINFO
PRINT FNAME
IF FNAME GT 'JAMES'
IF READLIMIT EQ 2
END
```

The report contains the following values for FNAME:

```
LARRY
JOHN
```

FOCUS considers the retrieval of a qualified segment a single read, even though IMS may read many segments to locate each one that it returns to FOCUS. In the previous example, IMS had to read three records in order to locate the two records that it returned to FOCUS.

In the next example, the requested value does not exist in the database but, because FNAME is qualified by an optimized screening condition, IMS searches the *entire* database trying to locate it:

```
TABLE FILE PATINFO
PRINT FNAME
IF FNAME GT 'TOMMY'
IF READLIMIT IS 1
END
```

To prevent this situation, the user should be aware of the valid range of values contained in the database when IMS search fields are part of a Master File.

4.4 The Dynamic JOIN Command

With the FOCUS dynamic JOIN command, you can reference two or more related databases or external files in a single FOCUS report request. The databases remain physically separate, but FOCUS treats them as a single logical structure.

The JOIN command joins a host file (the FROM file) to a cross-referenced file (the TO file) at execution time by matching values in one or more fields common to both files. IMS databases can participate as host or cross-referenced files in joins.

To issue a JOIN, you do not have to restructure the IMS database or rerun either the PSBGEN or the DBDGEN. However, when you join TO an IMS database, you must join to one of the following:

- A key field or secondary index in the root segment (ALIAS= IMSname.KEY, IMSname.HKY, or IMSname.SKY).
- The high-order portion of a key field or secondary index in the root segment.
- HDAM cannot use the high-order portion of a key field. It must be the full key.

Note: If a key consists of multiple fields, you must make it a GROUP in the Master File.

This restriction is necessary because FOCUS connects the databases at execution time using GET UNIQUE calls with a qualified SSA on the key field of the TO file.

You can join an IMS database to other IMS databases or, with the appropriate FOCUS Interfaces installed, you can join an IMS database to any other joinable database such as a DB2 table, a VSAM file, or a FOCUS database.

Two types of dynamic JOINS are available. The difference between the two depends on the cross-referenced file and its key values:

- **Multiple or non-unique join.** This defines a one-to-many or many-to-many correlation between the records of the host file and the records of the cross-referenced file. For each record in the host file, FOCUS may retrieve multiple matching records from the cross-referenced file.
- **Unique join.** This defines a one-to-one correlation between a record in the host file and one record in the cross-referenced file. For each record in the host file, FOCUS will retrieve at most one matching record from the cross-referenced file.

To create a non-unique join, include the ALL keyword in the JOIN command; for a unique join, omit the ALL keyword.

4.4.1 Single-Field Dynamic JOIN

To create a join based on relating one field in the host file to one field in the cross-referenced file, the syntax is

```
JOIN fielda1 [WITH rfield] IN hostfile [TAG tag]
  TO [ALL] fieldb1 IN crfile [TAG tag] [AS joinname]
  [END]
```

where:

<i>fielda1</i>	Is a field in the host file or a DEFINE field that contains values shared with <i>fieldb1</i> in the cross-referenced file.
WITH <i>rfield</i>	Is any real field in the same segment as the DEFINE field; associates the DEFINE field with a segment location in the host file. Use only if <i>fielda1</i> is a DEFINE field.
IN <i>hostfile</i>	Indicates the name of the host file. Use this name in subsequent report requests on the joined structure.
TAG <i>tag</i>	Indicates a one- to eight-character tag name (usually the Master File name). The tag name serves as a unique qualifier for fields and aliases in host or cross-referenced files.
TO ALL	Indicates that the host and cross-referenced files participate in a one-to-many or many-to-many relationship. In other words, for any value of the join field in the host file (<i>fielda1</i>), there may be more than one corresponding instance of that value for the join field in the cross-referenced file (<i>fieldb1</i>). This is known as a non-unique or multiple join. Omitting the ALL parameter indicates a unique join. Use only when each value of the join field in the host file (<i>fielda1</i>) has, at most, one possible corresponding value of the join field in the cross-referenced file (<i>fieldb1</i>).
<i>fieldb1</i>	Is a key, a secondary index, or the high-order portion of a key or secondary index in the root segment of the cross-referenced file; <i>fieldb1</i> shares values with <i>fielda1</i> . The ACTUAL format of <i>fieldb1</i> does not have to match the ACTUAL format of <i>fielda1</i> . However, the contents of <i>fielda1</i> must be convertible to a value of the same type as <i>fieldb1</i> .
IN <i>crfile</i>	Indicates the name of the cross-referenced file.
AS <i>joinname</i>	Assigns an optional one- to eight-character name to the join structure. Naming the join prevents subsequent JOIN commands from overwriting it and enables you to clear it selectively with the JOIN CLEAR command.
END	Required when the JOIN command is longer than one line; it terminates the statement.

Up to 16 JOIN commands can be active at one time. Joins operate in all environments.

Databases accessed by CMS through a Cross-Machine Interface can be joined together, to local CMS files, and to other databases accessed by another XMI.

Note:

- All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following error message:

```
(FOC4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```
- If you join an IMS database to itself (a recursive join), the PSB must contain two PCBs for the database.
- HSAM databases cannot participate as the cross-referenced (TO) file in a join because they lack an index or randomizing scheme for locating records.
- Except for HDAM, if you join a field in the host file to a longer field in the cross-referenced file, FOCUS implements the join as a partial key search and retrieves the records as in a range test (see Section 4.3.5, *Partial Key and Multi-Segment Requests*). Short to long joins are not supported against the primary key of an HDAM database since they result in incomplete record selection. However, short to long joins are supported if the join fields constitute the high-order portion of a secondary index.
- If you join a field in the host file to a shorter field in the cross-referenced file, FOCUS truncates the host field.
- When a record in the host file lacks a matching record in the cross-referenced file, the retrieval path is called a *short path*. Section 4.5, *Retrieval of Unique Segments*, illustrates an example. Refer to the *FOCUS for IBM Mainframe Users Manual* for a complete discussion of short path behavior.

The following examples illustrate the JOIN command

```
JOIN V1 IN VSAM1 TAG FILE1 TO I1 IN IMS1 TAG FILE2 AS JOIN1
JOIN I2 IN IMS2 TO ALL I3 IN IMS3 AS JOIN2
JOIN I1 IN IMS1 TO F1 IN FOCUS1 AS JOIN3
```

where:

VSAM1	Is a VSAM file.
IMS1 , IMS2 , IMS3	Are IMS databases.
FOCUS1	Is a FOCUS database.
V1	Is any field in VSAM1.
I1	Is a key to the root segment of IMS1.
I2	Is any field in IMS2.
I3	Is a key to the root segment of IMS3.
F1	Is a FOCUS indexed field.

4.4.2 Multi-Field Dynamic JOIN

To create a join based on relating multiple fields in the host and cross-referenced files, the syntax is

```
JOIN fielda1 AND fielda2 IN hostfile [TAG tag] TO [ALL]
   fieldb1 AND fieldb2 IN crfile [TAG tag] [AS joinname]
[END]
```

where:

<i>fielda1</i>	Is a field in the host file that shares values with <i>fieldb1</i> in the cross-referenced file.
AND <i>fielda2</i>	Indicates a field in the host file that shares values with <i>fieldb2</i> in the cross-referenced file.
IN <i>hostfile</i>	Indicates the name of the host file. Use this name in subsequent report requests on the joined structure.
TAG <i>tag</i>	Indicates a one- to eight-character tag name (usually the Master File name). The tag name serves as a unique qualifier for fields and aliases in host or cross-referenced files.
TO ALL	Indicates that the host and cross-referenced files participate in a one-to-many or many-to-many relationship. In other words, for any value of the join fields in the host file (<i>fielda1</i> , <i>fielda2</i>), there may be more than one corresponding instance of those values for the join fields in the cross-referenced file (<i>fieldb1</i> , <i>fieldb2</i>). This is known as a non-unique or multiple join. Omitting the ALL parameter indicates a unique join. Use only when each value of the join field in the host file (<i>fielda1</i>) has, at most, one possible corresponding value of the join field in the cross-referenced file (<i>fieldb1</i>).
<i>fieldb1</i>	Is a key, a secondary index, or the high-order portion of a key or secondary index in the root segment of the cross-referenced file; <i>fieldb1</i> shares values with <i>fielda1</i> . The ACTUAL format of <i>fieldb1</i> does not have to match the ACTUAL format of <i>fielda1</i> . However, the contents of <i>fielda1</i> must be convertible to a value of the same type as <i>fieldb1</i> .
<i>fieldb2</i>	Is a key, a secondary index, or the high-order portion of a key or secondary index in the root segment of the cross-referenced file; <i>fieldb2</i> shares values with <i>fielda2</i> . The ACTUAL format of <i>fieldb2</i> does not have to match the ACTUAL format of <i>fielda2</i> . However, the contents of <i>fielda2</i> must be convertible to a value of the same type as <i>fieldb2</i> .

Reporting Efficiencies

IN <i>crfile</i>	Indicates the name of the cross-referenced file.
AS <i>joinname</i>	Assigns an optional one- to eight-character name to the join structure. Naming the join prevents subsequent JOIN commands from overwriting it and enables you to selectively clear it with the JOIN CLEAR command.
END	Required when the JOIN command is longer than one line; it terminates the statement.

Up to 16 JOIN commands can be active at one time. Joins operate in all environments. Databases accessed by CMS through a Cross-Machine Interface can be joined together, to local CMS files, and to other databases accessed by another XMI.

Note:

- If you join TO multiple components of a key or secondary index (a *partial key join*), you must specify the ALL parameter to retrieve all matching records.
- All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following error message:

```
(FOC4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```
- If you join an IMS database to itself (a recursive join), the PSB must contain two PCBs for the database.
- HSAM databases cannot participate as the cross-referenced file in a join because they lack an index or randomizing scheme for locating records.
- Except for HDAM, if you join a field in the host file to a longer field in the cross-referenced file, FOCUS implements the join as a partial key search and retrieves the records as in a range test (see Section 4.3.5, *Partial Key and Multi-Segment Requests*). Short to long joins are not supported against the primary key of an HDAM database since they result in incomplete record selection. However, short to long joins are supported if the join fields constitute the high-order portion of a secondary index.
- If you join a field in the host file to a shorter field in the cross-referenced file, FOCUS truncates the host field.
- When a record in the host file lacks a matching record in the cross-referenced file, the retrieval path is called a *short path*. Section 4.5, *Retrieval of Unique Segments*, illustrates an example. Refer to the *FOCUS for IBM Mainframe Users Manual* for a complete discussion of short path behavior.

4.4.3 Join Optimization

To determine the best access path into the database, the Interface examines the record selection criteria in each report request (see Section 4.3.6, *Auto Index Selection*). When the request references a join, the Interface may use the results of this examination to retrieve the cross-referenced segment using a group key or secondary index.

Consider a single-field JOIN command in which the cross-referenced field is the high order portion of a group key or secondary index. Under certain conditions, the Interface uses the group key or secondary index to retrieve the cross-referenced segment, effectively converting the single-field join into a multi-field join. This conversion process is called *join optimization*.

The Interface can optimize a join when a TABLE or SQL Translator request satisfies the following two conditions:

- The report references a join on the high-order portion of a key or secondary index in the cross-referenced file.

For example, the EMPDB01 database contains fields SSNALPHA, EMPID, and LAST_NAME. In addition, the PATDB01 database has a secondary index, IXCOMP, that consists of fields SSNALPHA, EMPLOYEEID, and LAST_NAME (see Appendix A, *Sample File Descriptions*). The following request references a join on the high-order field of the IXCOMP secondary index:

```
JOIN SSNALPHA IN EMPDB01 TAG E1 TO SSNALPHA IN PATINFO TAG P1 AS J1
TABLE FILE EMPDB01
PRINT E1.SSNALPHA P1.SSNALPHA E1.EMPID P1.EMPLOYEEID
```

- The request includes optimizable record selection tests on one or more of the remaining high-order fields in the key or secondary index. For example:

```
WHERE E1.EMPID EQ P1.EMPLOYEEID
WHERE E1.LAST_NAME EQ P1.LAST_NAME
END
```

Auto Index Selection examines the request and determines that SSNALPHA, EMPLOYEEID, and LAST_NAME participate in a secondary index in the cross-referenced file. The following FSTRACE4 output illustrates that while the Interface generates an unqualified SSA to retrieve the EMPINFO segment, it retrieves the PATINFO segment using a qualified SSA on the IXCOMP index:

```
set up SSA-Q:
C5D4D7C9 D5C6D640 5C604040          *EMPINFO *-      *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DC9 E7C3D6D4  *PATINFO *- (IXCOM*
D740407E 40404040 40404040 40404040  *P =             *
40404040 40404040 40404040 40404040  *               *
40404040 40405D40          *           )      *
```

Reporting Efficiencies

If the request contains no record selection criteria against the low-order portion of the index, or it includes a range test on the low order portion of the index, the Interface constructs a qualified SSA with a range test. For example, the following request contains a greater-than test on the low-order field in the IXCOMP index:

```
JOIN SSNALPHA IN EMPDB01 TAG E1 TO SSNALPHA IN PATINFO TAG P1 AS J1
TABLE FILE EMPDB01
PRINT E1.SSNALPHA P1.SSNALPHA E1.EMPID P1.EMPLOYEEID
WHERE E1.EMPID EQ P1.EMPLOYEEID
WHERE E1.LAST_NAME GT P1.LAST_NAME
END
```

The qualified SSA that the Interface generates to retrieve the PATINFO segment incorporates a range test on the IXCOMP index:

```
set up SSA-Q:
C5D4D7C9 D5C6D640 5C604040          *EMPINFO *-      *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DC9 E7C3D6D4 *PATINFO *-(IXCOM*
D740406E 7E000000 00000000 00000000 *P  >=      *
00000000 00000000 00000000 00000000 *          *
00000000 00005CC9 E7C3D6D4 D740404C *          *IXCOMP <*
7EFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *=          *
FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *          *
FFFF5D40          * )          *
```

If an SQL Translator request contains an SQL SELECT statement that includes a join on the high-order portion of a key or secondary index, the Interface creates a qualified SSA using the appropriate key or secondary index.

For example, the following request joins the EMPDB01 database to the SSNALPHA, EMPLOYEEID, and LAST_NAME fields in the PATDB01 database:

```
SQL
SELECT E1.SSNALPHA, P1.SSNALPHA, E1.EMPID, P1.LAST_NAME, E1.LAST_NAME
FROM EMPDB01 E1, PATINFO P1
WHERE(E1.SSNALPHA = P1.SSNALPHA AND
E1.EMPID = P1.EMPLOYEEID AND
E1.LAST_NAME = P1.LAST_NAME);
END
```

The Interface retrieves the PATINFO segment using a qualified SSA on the IXCOMP index:

```
set up SSA-Q:
C5D4D7C9 D5C6D640 5C604040          *EMPINFO *-      *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DC9 E7C3D6D4 *PATINFO *-(IXCOM*
D740407E 40404040 40404040 40404040 *P  =        *
40404040 40404040 40404040 40404040 *          *
40404040 40405D40          *          )          *
```

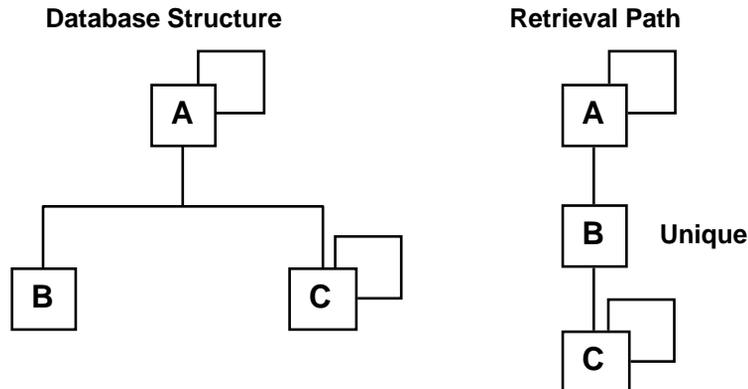
4.5 Retrieval of Unique Segments

When you define a segment in the IMS database as a unique descendant of its immediate parent, each instance of the parent can have at most one occurrence of that type of child. In the Master File, you identify a unique segment with `SEGTYPE=U`.

Normally, you define a segment as unique if the IMS segment is defined with no twin pointer (`PTR=NOTWIN`). However, since FOCUS has no way of verifying that the DBD describes the segment in this way, you can define a segment as unique whenever you want to retrieve only the first logical instance of that segment (per parent). This technique can be useful when the first instance is the current instance, and only current data from that segment is of interest.

FOCUS retrieves unique segments prior to retrieving any non-unique children of the same parent. You can see the order in which FOCUS retrieves segments by issuing the following command:

```
CHECK FILE filename PICTURE RETRIEVE
```



Reporting Efficiencies

Unique segments have two useful properties that reduce the complexity of the file:

- FOCUS views unique segments as logical extensions of their parents, so it never considers them missing.

When a report request references a non-unique child segment, FOCUS, by default, omits parent segments that have no instance of that child from the report. This retrieval path is called a short path. The retrieval of a unique child segment never results in a short path because FOCUS supplies default values for fields from absent unique children.

For example, in the preceding diagram, if the request references data from segment B, but segment A has no instance of segment B, the Interface generates suitable default values (blank if the field is alphanumeric, zero if numeric) to make it appear that a B instance was retrieved. Had segment B *not* been defined as unique, FOCUS would have omitted segment A from the report.

- The Interface always builds an SSA to retrieve unique segments. Notice that in the retrieval view, the file appears to have only one path.

These concepts also apply to joins. In a non-unique join, a host segment that lacks a matching cross-referenced segment represents a short path and is omitted from the resulting report. However, with a unique join, as with a unique segment, FOCUS supplies default values for missing cross-referenced fields in the resulting report.

For example, the following request references a unique join between two IMS databases that have no matching records:

```
JOIN SSN IN EMPDB01 TAG F1 TO SSNALPHA IN EMPDB02 TAG F2 AS J1
TABLE FILE EMPDB01
PRINT F1.SSN F2.SSNALPHA
END
```

Although the two files have no matching segments, the request produces a report that includes every segment from the host file; it assigns a blank value to each SSNALPHA field:

NUMBER OF RECORDS IN TABLE=		1667	LINES=		1667
PAGE		1			
	SSN	SSNALPHA			
	---	-----			
	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				
	9				
	10				
	.				
	.				
	.				

The next request is identical to the previous one, except for the keyword ALL in the JOIN command. Once again, there are no matching records in the host and cross-referenced databases:

```
JOIN SSN IN EMPDB01 TAG F1 TO ALL SSNALPHA IN EMPDB02 TAG F2 AS J1
TABLE FILE EMPDB01
PRINT F1.SSN F2.SSNALPHA
END
```

This request produces no report because the join is non-unique, and short paths are omitted:

NUMBER OF RECORDS IN TABLE=		0	LINES=		0
-----------------------------	--	---	--------	--	---

For a complete discussion of short path behavior, consult the *FOCUS for IBM Mainframe Users Manual*.

4.6 JOINS With Selection Criteria

In order to issue a TABLE request against joined IMS databases with selection criteria on the JOIN TO field, you must first issue the following command:

```
{MVS|TSO} IMS SET SSAOPT OFF
```

The IMS SET SSAOPT OFF command forces the Interface to include only the join criteria in the SSA. FOCUS then processes the other selection criteria.

For example:

```
MVS IMS SET SSAOPT OFF

JOIN SSN IN EMPDB01 TO SSN IN PATINFO AS J1

TABLE FILE EMPDB01
PRINT EMPDB01.SSN
IF PATINFO.SSN EQ ' '
END
```

This produces the following SSA:

```
set up SSA-Q:
C5D4D7C9 D5C6D640 5C604040          *EMPINFO *-      *
set up SSA-Q:
D7C1E3C9 D5C6D640 5C604DE2 E2D54040  *PATINFO *-(SSN *
4040407E 40404040 40404040 40405D40  * =      )      *
```

If you do not issue the IMS SET SSAOPT OFF command, then by default all selection criteria on keys will be passed to IMS for processing. This setting remains in effect for the entire session, unless you issue the IMS SET SSAOPT ON command.

5 Environments

IMS can run in several different environments and with various options. This chapter describes each environment and illustrates how to invoke the IMS/DB Interface from it. Regardless of which environment you use, the capabilities of the Interface are the same.

From the FOCUS perspective, three configurations are available:

- FOCUS can run in the same address space as an IMS component called the DBCTL stub and communicate directly with online IMS databases in a second address space. This configuration is available for IMS versions 3.1 and above; you implement it in your FOCUS session with SET commands. See Section 5.1, *Access to IMS Through DBCTL*, for information about the DBCTL environment.

This configuration does not support IMS access from CMS; it does support access from MSO.

- FOCUS can run in a separate address space from all IMS components and pass its DL/I calls to either the IMS region controller (DFSRRRC00) or the CICS region controller (DFHDRP) in a second address space.

To initiate this configuration, you execute either program DFSRRRC00 or program DFHDRP. This region controller program then loads the XMI server (program XMI). After the XMI server is loaded and executing, you can invoke the Interface from a CLIST or batch JCL. In order for the FOCUS address space to communicate with the region controller address space, you must allocate a common communication file in each. See Section 5.2, *Access to IMS Through the XMI Server*, for information about the XMI server environment.

Note:

- In prior releases, the BMP extension (program FOCBMP) served the function now performed by the XMI server. In Release 7.0, by default you access the XMI server, described in this chapter. For information about accessing the BMP extension, see Appendix C, *Release Dependent Interface Features*.
- IMS Version 5.1 does not support program DFHDRP. It supports access via DBCTL.

This is the required configuration for accessing IMS databases from CICS; for information about access from CICS, see Appendix C, *Release Dependent Interface Features*.

This configuration supports access from MSO.

Environments

- FOCUS can run in the same address space as the IMS region controller.

To invoke the Interface with this configuration, you execute either program DFSRRC00 (for IMS). This region controller then loads FOCUS.

Access to IMS databases with this configuration is quick, eliminates the separate FOCUS address space needed with the XMI server, and does not require you to allocate a communication file. However, it is not available through MSO or CMS. In addition, IMS or CICS must allow shared access to the databases. See Section 5.3, *Access to IMS With FOCUS Loaded by the Region Controller*, for information about this environment.

Note: IMS Version 5.1 does not support program DFHDRP.

Sections 5.1, *Access to IMS Through DBCTL*, 5.2, *Access to IMS Through the XMI Server*, and 5.3, *Access to IMS With FOCUS Loaded by the Region Controller*, discuss each environment and provide CLISTs and JCL for invoking the Interface. For your convenience, Section 5.4, *Summary Chart*, contains a chart that summarizes the environments available with the Interface, including the advantages and disadvantages of each. Section 5.5, *Environment Switching*, explains how you can change from one environment to another within your session.

Fast Path Considerations

Several of the environments through which you can invoke the Interface provide access to Fast Path databases (see the summary chart in Section 5.4, *Summary Chart*). If your site accesses Fast Path, it should have included certain parameters in the IMS control region startup procedure. You may need to know the values of the following parameters in order to access Fast Path through the Interface:

Parameter	Definition
BSIZ	Specifies the size of the Fast Path buffer. It should be large enough to handle the largest Fast Path record.
DBBF	Specifies the number of buffers (of size BSIZ) to be allocated. A formula for calculating DBBF follows this chart.
DBFX	Number of DBBFs IMS should page fix at startup time.

When you access Fast Path databases through the IMS region controller program, DFSRRC00, you specify two additional parameters: NBA (Normal Buffer Allocation, the number of DBBFs that the DFSRRC00 region can use) and OBA (Overflow Buffer Allocation, the number of DBBFs that the DFSRRC00 region can use as overflow buffers). Sections 5.2.1, *Initiating the XMI Server in BMP Mode: PARM='BMP,XMI,psbname'*, and 5.3.1, *Invoking the Interface in BMP Mode: PARM='BMP,FOCUS,psbname'*, discuss these parameters.

When you access Fast Path DEDB databases from the DBCTL environment, the CNBA parameter (included in the DRA Startup Table, discussed in Appendix D, *Installation Instructions*) identifies the number of Fast Path NBA buffers.

DBBF is determined by the following formula:

$$\text{DBBF} = \text{CNBA} + \text{NBA} + \text{OBA} + \text{DBFX} + (\text{a } 20\text{-}30\% \text{ cushion})$$

The PSB PROCOPT to Use With the Interface

Technically speaking, the Interface is not an IMS transaction, since it does not communicate through the IMS message queues and, therefore, does not require an IMSGEN. The PSB you use should specify PROCOPT=GO to inform IMS that the records returned to this program need not be enqueued; in the event of a failure of the IMS/DC region, the restart processing can ignore this program. PROCOPT=GO indicates read without integrity; that is, you may not get the most recent update.

Note, however, that the PROCOPT actually specified in the PSB, or in the PCB itself, is transparent to the Interface and has no effect on its retrieval logic. Therefore, you can actually use any PROCOPT. If you use PROCOPT=A, indicating all rights, you may get locked out while another user updates the database; also, IMS will use additional time and resources to check whether you updated the database, an operation not available through the Interface.

5.1 Access to IMS Through DBCTL

IMS versions 3.1 and above offer the DBCTL (Database Control) facility for accessing online IMS databases in MVS. DBCTL provides a number of security, efficiency, and application control enhancements and is easy to implement.

This section includes the following topics:

- Advantages of DBCTL (see Section 5.1.1, *Advantages of DBCTL*).
- How to invoke the Interface in the DBCTL environment (see Section 5.1.2, *Invoking the Interface in the DBCTL Environment*).
- How to implement DBCTL in your FOCUS session (see Section 5.1.3, *Implementing DBCTL*).

Note: DBCTL is not available for accessing IMS databases from CMS.

5.1.1 Advantages of DBCTL

The advantages of using DBCTL include the following:

- Direct communication between your application and IMS. Your task establishes its own connection to IMS by loading and calling the DBCTL subsystem when needed; it then issues DL/I calls directly to IMS. This implementation eliminates the need for a batch job (the XMI server), running in an additional address space, to intercept DL/I calls from FOCUS and transmit them to IMS.
- Enhanced security. You have access to standard security systems through the standard SAF interface (see Chapter 6, *Security*). The SAF interface is supported by security products such as RACF, CA-TOP SECRET, and CA-ACF2. Before allowing access to a particular PSB, the system verifies that the user is authorized to read the PSB.
- Dynamic PSB selection and switching within your application. In your JCL or CLIST, you allocate ddname FOCPSB to a partitioned dataset; then, within your application, you can issue a SET command to select any member PSB from the dataset.

You also have the option of creating Access Files that identify the appropriate PSB for each Master File (see Chapter 3, *Creating FOCUS Descriptions*). When you have Access Files, you do not have to issue a command to select a PSB, so the selection is transparent to your application.

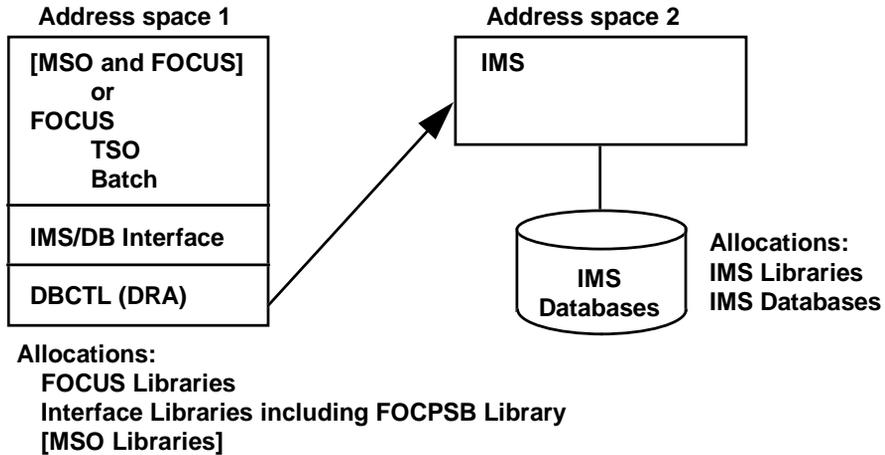
- Sharing of PCBs between applications. Multiple users can share the same PCBs, reducing the number of duplicate PCBs in a PSB and minimizing overhead.
- The ability to view current settings (PSB, IMSSEC, IMSCLASS, IMSMODE, IMSPZP) by issuing the IMS SET ? query command (see Appendix C, *Release Dependent Interface Features*, for an example).
- Access to most IMS databases (HDAM, HIDAM, Fast Path DEDB), except for Fast Path MSDB.

DBCTL must have been installed as described in Appendix D, *Installation Instructions*.

5.1.2 Invoking the Interface in the DBCTL Environment

The following diagram illustrates the relationship between FOCUS, the Interface, and IMS in the DBCTL environment:

MVS



Address Space	Contains
1	<ul style="list-style-type: none"> • Either MSO and FOCUS, or FOCUS without MSO. • The IMS/DB Interface. • The DBCTL stub, a component of IMS.
2	<ul style="list-style-type: none"> • The online IMS databases.

Environments

To invoke the Interface interactively within the DBCTL environment, edit the following CLIST to conform to your site's standards. In addition, add the IMS.RESLIB and the name of the PDS that contains the DRA Startup Table (load module DFSPZPxx, described in Appendix D, *Installation Instructions*) to the STEPLIB allocation in your TSO logon procedure. Note that the CLIST allocates the entire FOCPSB dataset, not just one member:

```
PROC 0
CONTROL NOMSG
FREE DDNAME(USERLIB ERRORS MASTER ACCESS FOCEXEC FOCPSB FOCLIB)
ALLOC F(ERRORS) DA('prefix.ERRORS.DATA') SHR REUSE
ALLOC F(USERLIB) DA('prefix.IMS.LOAD' +
                   'prefix.FUSELIB.LOAD') SHR REUSE
ALLOC F(MASTER) DA('prefix.MASTER.DATA') SHR REUSE
ALLOC F(ACCESS) DA('prefix.ACCESS.DATA') SHR REUSE
ALLOC F(FOCEXEC) DA('prefix.FOCEXEC.DATA') SHR REUSE
ALLOC F(FOCPSB) DA('prefix.FOCPSB.DATA') SHR REUSE
ALLOC F(FOCLIB) DA('prefix.FOCLIB.LOAD') SHR REUSE
CALL 'prefix.FOCLIB.LOAD(FOCUS)'
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

You can omit the allocation for ddname ACCESS if you select the psbname with the IMS SET PSB command, not in an Access File (see Section 5.1.3, *Implementing DBCTL*).

To invoke the Interface as a batch job, submit the following JCL after editing it to conform to your site's standards and adding a JOB card. Note that the JCL allocates the entire FOCPSB dataset, not just one member:

```
//FOCDBCTL EXEC PGM=FOCUS,REGION=nn
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
// DD DSN=IMS.RESLIB,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
// DD DSN=prefix.FUSELIB.LOAD,DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//FOCPSB DD DSN=prefix.FOCPSB.DATA,DISP=SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//MASTER DD DSN=prefix.MASTER.DATA,DISP=SHR
//ACCESS DD DSN=prefix.ACCESS.DATA,DISP=SHR
//SYSOUT DD SYSOUT=*
//*FSTRACE DD SYSOUT=*,DCB=BLKSIZE=133
//*FSTRACE4 DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
TSO IMS SET IMSPZP xx
TSO IMS SET DBCTL ON
TSO IMS SET PSB psbname
TABLE FILE ...
.
.
.
END
FIN
/*
//
```

where:

<i>nn</i>	Is the region size.
<i>prefix</i>	Is the high-level qualifier for your site's FOCUS production libraries.
<i>xx</i>	Is the suffix for the DRA Startup Table, chosen during installation (see Appendix D, <i>Installation Instructions</i>).
<i>psbname</i>	Is the name of the PSB to use.

Note:

- The SET commands in the sample are described in Section 5.1.3, *Implementing DBCTL*. You can omit the allocation for ddname ACCESS if you select the psbname with the IMS SET PSB command, as in the sample.
- IMS.RESLIB is the IMS load library, and 'prefix.IMS.LOAD' is the PDS that contains the DRA Startup Table (discussed in Appendix D, *Installation Instructions*) and the Interface load library.

5.1.3 Implementing DBCTL

To implement the DBCTL environment if you are *not* running MSO, issue the following commands in the order shown (if you are running MSO, these settings are in your configuration file, described in Appendix D, *Installation Instructions*):

```
{MVS|TSO} IMS SET IMSPZP {xx|00}
{MVS|TSO} IMS SET DBCTL ON
```

where:

xx Indicates the suffix for the DRA Startup Table, chosen during installation (see Appendix D, *Installation Instructions*). The default is 00.

Next, if you did not create an Access File to automatically select the PSB for your request (see Chapter 3, *Creating FOCUS Descriptions*), or to change the selected PSB, issue the following command either from the command line, in the MSOPROF profile, or in a FOCEXEC

```
{MVS|TSO} IMS SET PSB psbname
```

where:

psbname Is the name of the FOCPSB library member to use. This name must be identical to the name of the actual PSB that IMS will access. If the member does not exist, the following error message is generated:

```
(FOC4261) FOCPSB MEMBER NOT FOUND: psbname
```

Note:

- All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following error message:

```
(FOC4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```

- The IMS SET PSB command supersedes the PSB attribute in the Access File.

To see the settings in effect at any time during your session, issue the following query command:

```
{MVS|TSO} IMS SET ?
```

Appendix C, *Release Dependent Interface Features*, contains an example of the IMS SET ? query command.

5.2 Access to IMS Through the XMI Server

The XMI server is an application (program XMI) that intercepts DL/I calls from the IMS/DB Interface address space and issues them to IMS. Before you can invoke the Interface with this configuration, there must be an XMI server job executing with the appropriate parameter settings and with an available PCB for the databases you want to access.

Therefore, access to the XMI server is a two-part process. The two parts consist of the following:

1. Initiating an XMI server.

To initiate an XMI server, you run a region controller program and pass it parameters that describe the environment you need. The following chart lists the most common parameters:

Parameter	Value	Definition
Mode	BMP	Indicates Batch Message Processing mode. This mode accesses online IMS databases through the IMS/DC address space.
	DLI	Indicates local mode. This mode accesses IMS databases that you allocate locally.
Program	XMI	Loads and accesses the XMI server in the region controller address space. For information on accessing the BMP extension, see Appendix C, <i>Release Dependent Interface Features</i> .
PSB	psbname	Identifies the PSB to use.

One XMI server can service multiple users, but each user must have a separate PCB. Therefore, to allow multiple users, the PSB for the server should contain duplicate PCBs.

You may not have to actually execute this step because if an XMI server is already running with the correct parameters and an available PCB, you can use it.

2. Invoking the Interface once an XMI server job is running.

XMI server programs can be started at any time. In normal usage, one such job will be started in the morning with additional jobs started during the day, as the need for additional PCBs arises.

The job class and time expiration parameters used with the XMI server job must allow for a long residence in the system, and the priority should be sufficiently high to give good response to the users being serviced. In setting the various job parameters that affect response, remember that these jobs spend most of their time in the WAIT state consuming no resources, since they have nothing to do when not responding to DL/I calls. Even when processing such calls, they only do minimal work. All the logical work of the Interface is done in the various TSO or MSO address spaces.

When you use the XMI server to access IMS databases, you must allocate a common communication file (to ddname FOCBMP) in both the FOCUS address space and the XMI server address space. The Communication File is discussed in Section 5.2.5, *Additional Instructions for Using the XMI Server*.

This section includes the following topics:

- How to initiate an XMI server for access to online databases allocated in the IMS/DC region (see Section 5.2.1, *Initiating the XMI Server in BMP Mode: PARM='BMP,XMI,psbname'*).
- How to initiate an XMI server for access to databases allocated locally (see Section 5.2.2, *Initiating the XMI Server in DLI Mode: PARM='DLI,XMI,psbname'*).
- How to initiate an XMI server for access to online databases under the control of CICS (see Section 5.2.3, *Initiating the XMI Server With Databases Under Control of CICS*).
- How to invoke the Interface once an XMI server has been initiated (see Section 5.2.4, *Invoking the Interface in the XMI Server Environment*).
- Additional instructions for the XMI server environment, including allocation of the communication file, characteristics of PSBs for the XMI server, and termination of an XMI server (see Section 5.2.5, *Additional Instructions for Using the XMI Server*).

If your site does not have IMS 3.1 (or a subsequent version) installed, you must use the XMI server to access IMS databases if either of the following conditions exists:

- You are using MSO.
- IMS/DC or CICS has exclusive control of the IMS databases and does not permit other jobs in the system to allocate them, even in SHR mode.

Because this situation is common, the use of the IMS/DB Interface without the XMI server is largely limited to batch jobs that can allocate the required databases (because IMS/DC or CICS is not active) or to single users.

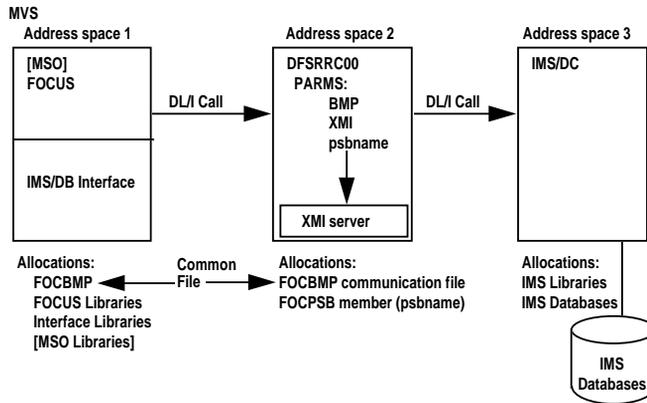
If IMS/DC or CICS *does* permit concurrent database access by other jobs, and you are not running from MSO or CMS, you can run FOCUS without the XMI server, as described in Section 5.3, *Access to IMS With FOCUS Loaded by the Region Controller*.

The following sections provide detailed information about the XMI server, including sample JCL and CLISTS.

5.2.1 Initiating the XMI Server in BMP Mode: PARM='BMP,XMI,psbname'

BMP (Batch Message Processing) mode accesses online IMS databases under the control of IMS/DC. Therefore, you can access Fast Path databases with this configuration, and, if a record is updated, you retrieve the updated version.

The following diagram illustrates the relationship between FOCUS, the XMI server, and IMS:



Address Space	Contains
1	<ul style="list-style-type: none"> • Either MSO and FOCUS, or FOCUS without MSO. • The IMS/DB Interface. • No IMS components. To communicate with the XMI server, you must allocate a communication file.
2	<ul style="list-style-type: none"> • The IMS region controller, DFSRR00. • The XMI server. <p>DFSRR00 loads the XMI server into its address space because of the XMI parameter in your JCL. You must allocate the same communication file here as in the FOCUS address space.</p> <p>You must select the PSB prior to your session and pass its name to program DFSRR00 as a parameter. This member is the sequential file you allocate to ddname FOCPSB.</p>
3	<ul style="list-style-type: none"> • IMS/DC. <p>The BMP parameter that you pass to program DFSRR00 indicates that the IMS databases are online and are allocated in the IMS/DC address space.</p>

Environments

To initiate an XMI server with this configuration, submit the following JCL after editing it to conform to your site's standards and adding a JOB card.

You do not have to submit this JCL if an XMI server job that has an available PCB is already running with the following parameter settings

```
//BMPXMI EXEC PGM=DFSRR00,REGION=nn,PARM='BMP,XMI,psbname[,,,,,,nba,oba]'
//STEPLIB DD DSN=IMS.RESLIB,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
// DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
//FOCPSB DD DSN=prefix.FOCPSB(psbname),DISP=SHR
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//
```

where:

nn Is the region size.

psbname Is the name of the PSB to use.

nba Is the number of buffers for the normal buffer allocation. It is required only for access to Fast Path databases; omit it if you will not access Fast Path.

Note: There are nine commas between the *psbname* and *nba* parameters.

oba Is the number of buffers for the overflow buffer allocation. It is required only for access to Fast Path databases; omit it if you will not access Fast Path.

If you omit the *nba* and *oba* parameters and attempt to access Fast Path databases, the following message is generated:

```
(FOC4214) REMOTE DLI CALL ERROR STATUS FOR SEGMENT:XXX/ FR
```

FR is an IMS status code indicating that too few buffers were allocated.

prefix Is the high-level qualifier for your site's FOCUS production libraries.

Note:

- IMS.RESLIB is the IMS load library.
- Ddname FOCBMP is allocated to the common communication file. You must allocate the same file when you invoke the Interface, as described in Section 5.2.4, *Invoking the Interface in the XMI Server Environment*.
- Most error conditions detected by the XMI server are reflected back to the originating user, where they are issued to the terminal. The SYSPRINT file allocated in the XMI server job will contain only global error messages pertaining to the XMI server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the XMI server are discussed in Section 5.2.5, *Additional Instructions for Using the XMI Server*.

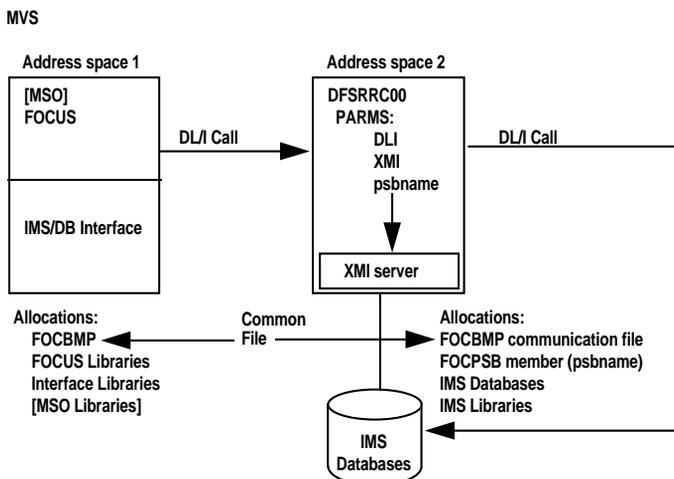
Once the XMI server job is executing, you can invoke the Interface from a CLIST or from batch JCL, as described in Section 5.2.4, *Invoking the Interface in the XMI Server Environment*.

5.2.2 Initiating the XMI Server in DLI Mode: PARM='DLI,XMI,psbname'

DLI mode accesses IMS databases that you allocate locally in the DFSRRC00 address space. Since you do not access the IMS/DC address space, you cannot use Fast Path databases and, if a record is updated during your session, you cannot retrieve the updated record.

Note: If you substitute the parameter DBB for DLI, IMS uses ACBs when accessing databases. This ensures that the database is in sync with the IMS descriptions you are using.

The following diagram illustrates the relationship between FOCUS, the XMI server, and IMS:



Address Space	Contains
1	<ul style="list-style-type: none"> • Either MSO and FOCUS, or FOCUS without MSO. • The IMS/DB Interface. • No IMS components. To communicate with the XMI server, you must allocate a communication file.
2	<ul style="list-style-type: none"> • The IMS region controller, DFSRRC00. • The XMI server. <p>DFSRRC00 loads the XMI server into its address space because of the XMI parameter in your JCL. You must allocate the same communication file here as in the FOCUS address space.</p> <p>The DLI parameter that you pass to program DFSRRC00 indicates that the IMS databases are allocated locally in the DFSRRC00 address space.</p> <p>You must select the PSB prior to your session and pass its name to program DFSRRC00 as a parameter. This member is the sequential file you allocate to ddname FOCPSB.</p>

To initiate an XMI server with this configuration, submit the following JCL after editing it to conform to your site's standards and adding a JOB card. Note that you must allocate your IMS databases, DBDs, and PSBs in this JCL.

You do not have to submit this JCL if an XMI server job that has an available PCB is already running with the following parameter settings

```
//DLIXMI EXEC PGM=DFSRRC00,REGION=nn,PARM='DLI,XMI,psbname'
//STEPLIB DD DSN=IMS.RESLIB,DISP=SHR
// DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//IMS DD DSN=IMS.PSBLIB,DISP=SHR
// DD DSN=IMS.DBDLIB,DISP=SHR
//DFSVSAMP DD DSN=IMS.DFSVSAMP(bufpool),DISP=SHR
//PATDB01 DD DSN=IMS.PATIENT.DB,DISP=SHR
//PATDBIX DD DSN=IMS.PATIENT.IX,DISP=SHR
//PATDBIX1 DD DSN=IMS.PATIENT.IX1,DISP=SHR
//PATDBIX2 DD DSN=IMS.PATIENT.IX2,DISP=SHR
//PATDBIX3 DD DSN=IMS.PATIENT.IX3,DISP=SHR
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
//FOCPSB DD DSN=prefix.IMS.FOCPSB(psbname),DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//
```

where:

<i>nn</i>	Is the region size.
<i>psbname</i>	Is the name of the PSB to use.
<i>prefix</i>	Is the high-level qualifier for your site's FOCUS production libraries.
<i>bufpool</i>	Is the member that contains your VSAM buffer pool information.

Note:

- The ddnames allocated to the IMS databases are from the DD1 parameters in the relevant IMS DBDs. In the sample, the databases are the PATDB databases and indexes described in Appendix A, *Sample File Descriptions*.
- IMS.RESLIB is the IMS load library, IMS.DBDLIB is the IMS DBD library, and IMS.PSBLIB is the IMS PSB library.
- If you specify the parameter DBB instead of DLI on the EXEC card, you must include the ACB library in the allocation for ddname IMS. In addition, you must allocate a second dataset, IMS.ACBLIB, to ddname IMSACB.
- Ddname FOCBMP is allocated to the common communication file. You must allocate the same file when you invoke the Interface, as described in Section 5.2.4, *Invoking the Interface in the XMI Server Environment*.
- Most error conditions detected by the XMI server are reflected back to the originating user, where they are issued to the terminal. The SYSPRINT file allocated in the XMI server job will contain only global error messages pertaining to the XMI server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the XMI server are discussed in Section 5.2.5, *Additional Instructions for Using the XMI Server*.

Once the XMI server job is executing, you can invoke the Interface from a CLIST or from batch JCL, as described in Section 5.2.4, *Invoking the Interface in the XMI Server Environment*.

5.2.3 Initiating the XMI Server With Databases Under Control of CICS

If you are using CICS Version 3.3 or lower, see the instructions for *Changing Addressability Mode of the XMI Server* in the next section.

If you are using a CICS Version later than Version 3.3, skip the next section and go to the section entitled *Program DFHDRP*.

Changing Addressability Mode of the XMI Server

In CICS Version 3.3 and lower, the module DFSDRP (CICS version of the IMS region controller) ran in a 24-bit addressability mode. Our XMI Server, which is invoked by the module DFHDRP, executes in a 31-bit addressability mode. As a result, the DFHDRPE module abends with an SOC4.

The following JCL will allow you to change the XMI Server from passing data in 31-bit mode to 24-bit mode. It is included on the tape in prefix.CTL LIB.DATA(IMTDLI2).

```

/*-----
/* THIS JCL WILL ALLOW THE XMI SERVER TO ISSUE DLI CALLS TO CBLTDLI
/* WHEN THE DLI SYSTEM IS AMODE 24 (EG. CICS330)
/* PLEASE UPDATE THE DATASET NAMES TO REFLECT THOSE ON YOUR SYSTEM
/*-----
/*
//LNKIMSRV EXEC PGM=IEWL,PARM='NCAL,LET,LIST,SIZE=1024K'
//OLDMOD DD DSN=prod.FOCLIB.LOAD,DISP=SHR
//FOCCTL DD DSN=prod.FOCCTL.DATA,DISP=SHR
//IMTDLI2 DD DSN=prod.FOCCTL.DATA,(IMTDLI2),DISP=SHR
//SYSLMOD DD DSN=new.FOCLIB.LOAD,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//SYSPRINT DD SYSOUT=*
//SYSLIN DD
INCLUDE IMTDLI2
INCLUDE OLDMOD (IMSRV)
INCLUDE FOCCTL (IMSRV)
NAME IMSRV (R)
/*

```

Modify the JCL and submit it. Upon successful completion, the XMI Server will function in 24-bit mode which is compatible with the CICS release.

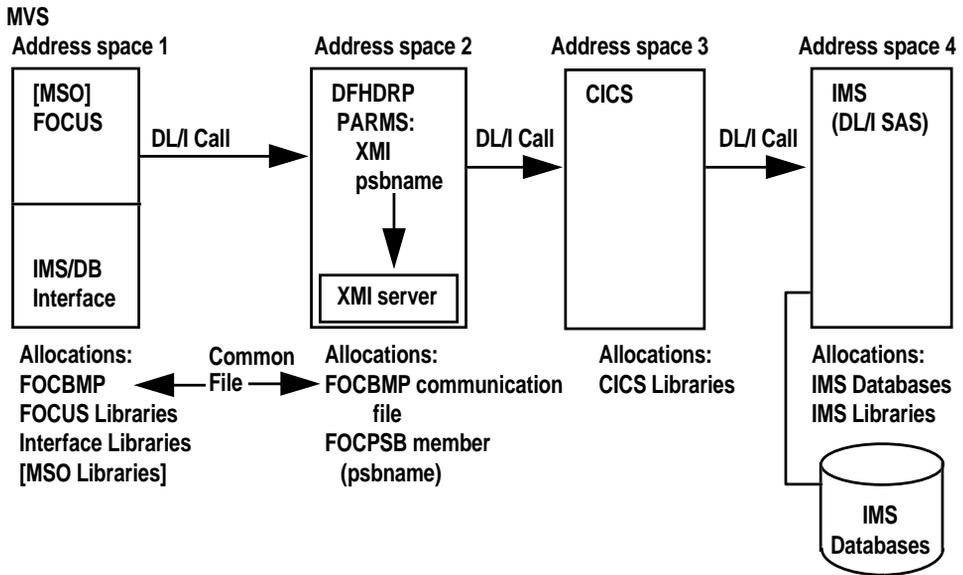
Program DFHDRP

Program DFHDRP accesses online IMS databases under the control of CICS. This configuration consumes a lot of communication resources and, since there is no IMS/DC address space, it cannot access Fast Path databases. However, since the databases are online, if a record is updated, you retrieve the updated version.

Note:

- You should use this configuration only when CICS controls the databases in non-share mode. If your site has DBRC and sharing is allowed, use the configuration described in Section 5.2.1, *Initiating the XMI Server in BMP Mode: PARM= 'BMP,XMI,psbname'*.
- IMS Version 5.1 does not support this configuration. You must use DBCTL instead (see Section 5.1, *Access to IMS Through DBCTL*).

The next diagram illustrates the relationship between FOCUS, the XMI server, CICS, and IMS:



Address Space	Contains
1	<ul style="list-style-type: none"> • Either MSO and FOCUS, or FOCUS without MSO. • The IMS/DB Interface. • No IMS components. To communicate with the XMI server, you must allocate a communication file.

Address Space	Contains
2	<ul style="list-style-type: none"> • The CICS region controller, DFHDRP. • The XMI server. <p>DFHDRP loads the XMI server into its address space because of the XMI parameter in your JCL. You must allocate the same communication file here as in the FOCUS address space.</p> <p>You must select the PSB prior to your session and pass its name to program DFHDRP as a parameter. This member is the sequential file you allocate to ddname FOCPSB.</p>
3	<ul style="list-style-type: none"> • CICS.
4	<ul style="list-style-type: none"> • IMS DL/I SAS (Separate Address Space). <p>The IMS databases are online and are allocated in the IMS DL/I SAS address space.</p>

To initiate an XMI server with this configuration, submit the following JCL after editing it to conform to your site's standards and adding a JOB card.

You do not have to submit this JCL if an XMI server job that has an available PCB is already running with the following parameter settings

```

//XMICICS EXEC PGM=DFHDRP,REGION=nn,PARM='SSA=3000,PGM=XMI,PSB=psbname,
//          CICS=cics_applid'
//STEPLIB DD DSN=CICS.LOAD,DISP=SHR
//          DD DSN=prefix.IMS.LOAD,DISP=SHR
//          DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//DFHLIB DD DSN=CICS.LOAD,DISP=SHR
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
//FOCPSB DD DSN=prefix.FOCPSB(psbname),DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//

```

where:

- nn* Is the region size.
- psbname* Is the name of the PSB to use.
- cics_applid* Is the VTAM application ID for the CICS address space you will access.
- prefix* Is the high-level qualifier for your site's FOCUS production libraries.

Note:

- CICS.LOAD is the CICS load library.
- Ddname FOCBMP is allocated to the common communication file. You must allocate the same file when you invoke the Interface, as described in Section 5.2.4, *Invoking the Interface in the XMI Server Environment*.
- Most error conditions detected by the XMI server are reflected back to the originating user, where they are issued to the terminal. The SYSPRINT file allocated in the XMI server job will contain only global error messages pertaining to the XMI server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the XMI server are discussed in Section 5.2.5, *Additional Instructions for Using the XMI Server*.

Once the XMI server job is executing, you can invoke the Interface from a CLIST or from batch JCL, as described in Section 5.2.4, *Invoking the Interface in the XMI Server Environment*.

5.2.4 Invoking the Interface in the XMI Server Environment

Before you can invoke the Interface in the XMI server environment, an XMI server job, initiated with the appropriate parameter settings, must be running and must have an available PCB for each database you will access. PCBs for the XMI server are discussed in Section 5.2.5, *Additional Instructions for Using the XMI Server*.

To invoke the Interface interactively, execute the following CLIST after editing it to conform to your site's standards

```
PROC 0
CONTROL NOMSG
FREE DDNAME(USERLIB ERRORS MASTER FOCEXEC FOCBMP FOCLIB)
ALLOC F(ERRORS) DA('prefix.ERRORS.DATA') SHR REUSE
ALLOC F(USERLIB) DA('prefix.IMS.LOAD' +
'prefix.FUSELIB.LOAD') SHR REUSE
ALLOC F(MASTER) DA('prefix.MASTER.DATA') SHR REUSE
ALLOC F(FOCEXEC) DA('prefix.FOCEXEC.DATA') SHR REUSE
ALLOC F(FOCBMP) DA('prefix.FOCBMP.DATA') SHR REUSE
ALLOC F(FOCLIB) DA('prefix.FOCLIB.LOAD') SHR REUSE
CALL 'prefix.FOCLIB.LOAD(FOCUS)'
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

Environments

To invoke the Interface as a batch job, submit the following sample JCL after editing it to conform to your site's standards and adding a JOB card

```
//FOCXMI EXEC PGM=FOCUS,REGION=nn
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
// DD DSN=prefix.FUSELIB.LOAD,DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//MASTER DD DSN=prefix.MASTER.DATA,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
TABLE FILE ...
.
.
.
END
FIN
/*
//
```

where:

nn Is the region size.

prefix Is the high-level qualifier for your site's FOCUS production libraries.

5.2.5 Additional Instructions for Using the XMI Server

This section explains how to allocate the common communication files needed for the XMI server, how to create PSBs for use with the XMI server, and how to terminate XMI server jobs.

The Communication File

The “handshake” between the Interface, running in its address space (or as a batch job), and XMI, running in another address space, requires a communication file accessible to both. This file must be allocated to ddname FOCBMP. It consists of a single 16-byte record, and it plays no role in subsequent message traffic between the two programs. Space for this file must be allocated once (on a permanently mounted disk pack) and catalogued as part of the installation of the XMI server (see Appendix D, *Installation Instructions*). The dataset name is irrelevant.

Note: The following discussion assumes that the FOCUS production libraries are stored in datasets catalogued under the same high-level qualifier. The identifier “prefix” denotes this common high-level qualifier.

After the communication file has been created, when you subsequently invoke the Interface and the XMI server, allocate this file as follows:

- In the JCL that initiates the XMI server:

```
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
```

- In the FOCUS application:

From	Allocation
The CLIST that invokes FOCUS	ALLOC F(FOCBMP) DA('prefix.FOCBMP.DATA') SHR REUSE
A FOCUS session	{MVS TSO} ALLOC F(FOCBMP) DA('prefix.FOCBMP.DATA') SHR REU
Batch FOCUS JCL or MSO server JCL	//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
An MSO or FOCUS session	DYNAM ALLOC FILE FOCBMP DATASET prefix.FOCBMP.DATA SHR

When running MSO, it is recommended that you allocate the communication files in the MSO server JCL. From TSO, you can allocate the FOCBMP file any time before you first attempt to use it, in the logon procedure, from a CLIST executed before you enter FOCUS, or from within the session.

When several concurrent XMI jobs are active, each one requires its own communication file. Each XMI server job must allocate a different file—it does not matter which one—to ddname FOCBMP.

For example, XMI job 1:

```
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
```

XMI job 2:

```
//FOCBMP DD DSN=prefix.FOCBMP1.DATA,DISP=SHR
```

Every TSO ID or MSO server must allocate *all* of the communication files corresponding to the XMI jobs it might want to use. Each communication file must be allocated to a different but standard ddname. These ddnames are FOCBMP, and FOCBMP1 through FOCBMP15. For example:

```
DYNAM ALLOC FILE FOCBMP DA prefix.FOCBMP.DATA SHR
DYNAM ALLOC FILE FOCBMP1 DA prefix.FOCBMP1.DATA SHR
```

It does not matter how the communication files and the FOCBMP ddnames are paired. For example, if two XMI server jobs are active, the user can allocate their files to any two of the sixteen standard ddnames.

In the interest of simplicity, each user should allocate all of the communication files that were created at installation time, even if the corresponding XMI jobs may not be active. This technique makes it possible to use a common CLIST for these allocations.

If none of the FOCBMP files are allocated, the Interface does not attempt to use the XMI server; it issues its own DL/I calls instead. These DL/I calls will be successfully transmitted to IMS only if you are not running MSO and if your environment is configured as described in Section 5.3, *Access to IMS With FOCUS Loaded by the Region Controller*.

PSBs for the XMI Server

Since IMS uses the PCB to maintain an application program's position in the database, concurrent users who are serviced by the same XMI job cannot share PCBs.

Each user application that accesses the server queries all the XMI jobs to which it is linked through the communication files, until it finds one that has a free PCB appropriate for reporting on the database. (The search for an available PCB is transparent to the user.) The PCB remains occupied for the duration of the retrieval portion of a request and is freed prior to the output portion of the request.

Thus, if ten users need concurrent access to the SALES database, the PSB must contain at least ten PCBs that correspond to the Master File for SALES. The eleventh user will receive a diagnostic message indicating that there is no free PCB available at the time. (In this case, the user should initiate another XMI server, as illustrated in previous sections.)

The ten SALES PCBs can reside in one of the following:

- In a single PSB that requires only one XMI job.
- In ten identical PSBs with one SALES PCB each for ten concurrent XMI jobs.
- In any other combination that adds up to ten.

The IMS DBA must determine which configuration is best for the site. The tradeoff is between a single large PSB that ties up only one XMI region or several XMI regions, each running a non-duplicated and therefore smaller PSB.

Note: If the PSB contains multiple PCBs for the same database, in order to allow access through secondary indexes, you must duplicate the whole block of PCBs. Chapter 2, *IMS Overview and Mapping Concepts*, and Chapter 3, *Creating FOCUS Descriptions*, describe secondary index PCBs.

Terminating an XMI Server

A successfully started XMI server job will not end automatically; the operator must terminate it in one of three ways:

- By executing job XMIKILL, described in this section. This is the recommended termination method.
- Through an MVS cancel.

The MVS cancel should be avoided because of its potential impact on the IMS/DC region if the cancellation occurs while the XMI server is processing a DL/I call. The MVS cancel will also bypass the server's own cleanup process with the result that a 4K area in the CSA (Common Storage Area) will not be freed and will remain unavailable until the next IPL.

- Through an IMS cancel.

The IMS cancel will properly disconnect the XMI server and IMS/DC regions without any impact on the latter, but it will bypass the freeing of the 4K communication area in the CSA.

Termination through job XMIKILL avoids problems. The job sends the server a termination message that breaks into the server message queue ahead of all other messages addressed to it. Upon receipt of this message, the server performs its cleanup and returns to IMS, signaling a normal completion.

Sample JCL for job XMIKILL follows

```
//XMIKILL EXEC PGM=FOCUS,REGION=nn
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//SYSOUT DD SYSOUT=*
//ddname DD DSN=comfile,DISP=SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
EX XMI ACTION=STOP,GATEWAY=FOCBMP
FIN
/*
//
```

where:

<i>nn</i>	Is the region size.
<i>prefix</i>	Is the high-level qualifier for your site's FOCUS production libraries.
<i>ddname</i>	Is the ddname allocated to the communication file of the XMI server job to be terminated. If only one job is to be terminated, use ddname FOCBMP.
<i>comfile</i>	Is the dataset name of the communication file that is allocated to the XMI server job you want to terminate; for example, 'prefix.FOCBMP.DATA'.

Any of the three termination methods can be invoked at any time, without impact on those who might be actively using the canceled job (except for the impact on IMS/DC discussed previously). All such users will receive one of several error messages, depending on where in their retrieval process the cancellation occurred. Their retrieval ends cleanly; in no case will the user ABEND or be left with a hung terminal.

5.3 Access to IMS With FOCUS Loaded by the Region Controller

When you access IMS through the XMI server, each DL/I call requires communication resources. If you are not using MSO or CMS, and if IMS or CICS allows shared access to the IMS databases, it is more efficient to eliminate the XMI server by having the region controller load FOCUS, rather than the XMI server, into its address space. This section includes the following topics:

- How to invoke the Interface with online IMS databases that are allocated in the IMS/DC region (see Section 5.3.1, *Invoking the Interface in BMP Mode: PARM='BMP,FOCUS,psbname'*).
- How to invoke the Interface with IMS databases that are allocated locally (see Section 5.3.2, *Invoking the Interface in DLI Mode: PARM='DLI,FOCUS,psbname'*).

Note: If you have IMS 3.1 or above, you can access IMS through DBCTL (see Section 5.1, *Access to IMS Through DBCTL*).

To invoke FOCUS in this configuration, execute the IMS region controller program (DFSRRRC00) or the CICS region controller program (DFHDRP), depending on whether IMS/DC or CICS has control of the IMS databases. (You can also call the region controller from a CLIST in this configuration.) You must pass the program a list of parameters that define the environment you need. The following chart lists the most common parameters:

Parameter	Value	Definition
Mode	BMP	Indicates Batch Message Processing mode. This mode accesses online IMS databases through the IMS/DC address space.
	DLI	Indicates local mode. This mode accesses IMS databases that you allocate locally.
Program	FOCUS	Loads and accesses FOCUS in the region controller address space.
PSB	psbname	Identifies the PSB to use.

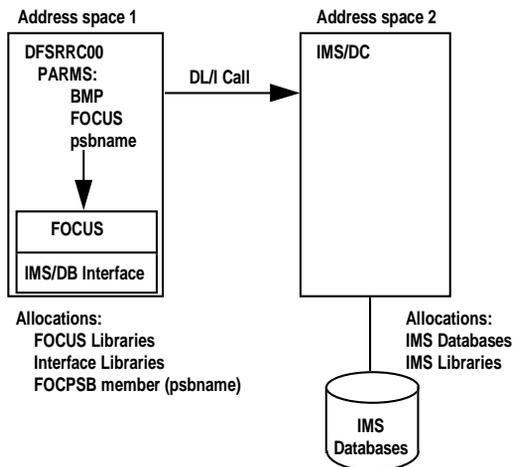
The following sections provide detailed information, including CLISTs and JCL.

5.3.1 Invoking the Interface in BMP Mode: PARM='BMP,FOCUS,psbname'

BMP mode accesses online IMS databases under the control of IMS/DC. Therefore, you can access Fast Path databases with this configuration, and, if a record is updated, you retrieve the updated version.

The following diagram illustrates the relationship between FOCUS, the IMS region controller, and IMS/DC:

MVS



Address Space	Contains
1	<ul style="list-style-type: none"> The IMS region controller (DFSRRRC00). FOCUS. The IMS/DB Interface. <p>DFSRRRC00 loads FOCUS into its address space because of the FOCUS parameter in your JCL or CLIST. There is no allocation for a communication file.</p> <p>You must select the PSB prior to your session and pass its name to program DFSRRRC00 as a parameter. This member is the sequential file you allocate to ddname FOCPSB.</p>
2	<ul style="list-style-type: none"> IMS/DC. <p>The BMP parameter that you pass to program DFSRRRC00 indicates that the IMS databases are online and are allocated in the IMS/DC address space.</p>

Environments

To invoke the Interface interactively from this environment, first add IMS.RESLIB and 'prefix.FOCLIB.LOAD' to the allocation for ddname STEPLIB in your TSO logon procedure. Execute the following CLIST after editing it to conform to your site's standards

```
PROC 0
CONTROL NOMSG
FREE DDNAME(FOCP SB MASTER FOCEXEC ERRORS USERLIB)
ALLOC F(DFSRESLB) DA('IMS.RESLIB') SHR REUSE
ALLOC F(MASTER) DA('prefix.MASTER.DATA') SHR REUSE
ALLOC F(FOCEXEC) DA('prefix.FOCEXEC.DATA') SHR REUSE
ALLOC F(FOCP SB) DA('prefix.FOCP SB.DATA(psbname)') SHR REUSE
ALLOC F(ERRORS) DA('prefix.ERRORS.DATA') SHR REUSE
ALLOC F(USERLIB) DA('prefix.IMS.LOAD' +
                    'prefix.FUSELIB.LOAD') SHR REUSE
CALL 'IMS.RESLIB(DFSRR C00)' 'BMP, FOCUS, psbname[ , , , , , , , , , nba, oba]'
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

psbname Is the name of the PSB to use.

nba Is the number of buffers for the normal buffer allocation. It is required only for access to Fast Path databases; omit it if you will not access Fast Path.

Note: There are nine commas between the psbname and nba parameters.

oba Is the number of buffers for the overflow buffer allocation. Is required only for access to Fast Path databases; omit it if you will not access Fast Path.

If you omit the nba and oba parameters and attempt to access Fast Path databases, the following message is generated:

```
(FOC4214) REMOTE DLI CALL ERROR STATUS FOR SEGMENT:XXX /FR
```

FR is an IMS status code indicating that too few buffers were allocated.

Access to IMS With FOCUS Loaded by the Region Controller

To invoke the Interface as a batch job, submit the following JCL after editing it to conform to your site's standards and adding a JOB card

```
//STEP1 EXEC PGM=DFSRR00,REGION=nn,PARM='BMP,FOCUS,psbname[,,,,,,,,,nba,oba]'
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
// DD DSN=IMS.RESLIB,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
// DD DSN=prefix.FUSELIB.LOAD,DISP=SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//MASTER DD DSN=prefix.MASTER.DATA,DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//FOCP SB DD DSN=prefix.FOCP SB.DATA(psbname),DISP=SHR
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//SYS PRINT DD SYSOUT=*
//OFFLINE DD SYSOUT=*
//SYSIN DD *
TABLE FILE ...
.
.
.
END
FIN
/*
//
```

where:

nn Is the region size.

psbname Is the name of the PSB to use.

nba Is the number of buffers for the normal buffer allocation. It is required only for access to Fast Path databases; omit it if you will not access Fast Path.

Note: There are nine commas between the *psbname* and *nba* parameters.

oba Is the number of buffers for the overflow buffer allocation. It is required only for access to Fast Path databases; omit it if you will not access Fast Path.

If you omit the *nba* and *oba* parameters and attempt to access Fast Path databases, the following message is generated:

```
(FOC4214) REMOTE DLI CALL ERROR STATUS FOR SEGMENT:XXX /FR
```

FR is an IMS status code indicating that too few buffers were allocated.

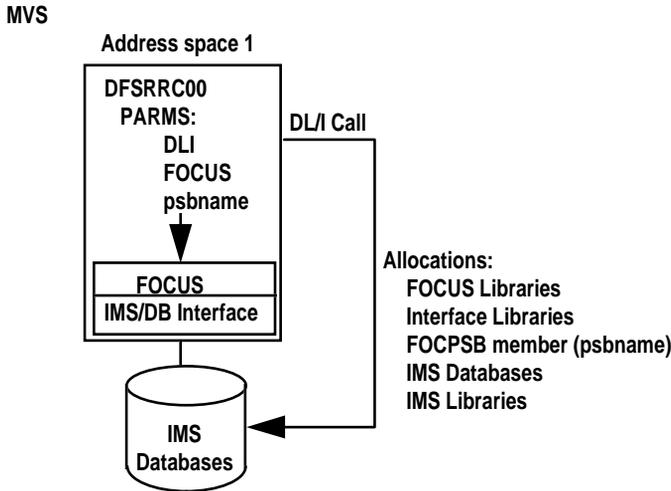
prefix Is the high-level qualifier for your site's FOCUS production libraries.

5.3.2 Invoking the Interface in DLI Mode: PARM='DLI,FOCUS,psbname'

DLI mode accesses IMS databases that you allocate locally in the DFSRRC00 address space. Since you do not access the IMS/DC address space, you cannot use Fast Path databases and, if a record is updated during your session, you cannot retrieve the updated record.

Note: If you substitute the parameter DBB for DLI, IMS uses ACBs to access its databases.

The next diagram shows the relationship between FOCUS, the IMS region controller, and IMS:



Address Space	Contains
1	<p>The only address space in this configuration contains the following:</p> <ul style="list-style-type: none"> • The IMS region controller (DFSRRC00). • FOCUS. • The IMS/DB Interface. <p>DFSRRC00 loads FOCUS into its address space because of the FOCUS parameter in your JCL or CLIST. There is no allocation for a communication file.</p> <p>The DLI parameter that you pass to program DFSRRC00 indicates that the IMS databases are allocated locally in the DFSRRC00 address space.</p> <p>You must select the PSB prior to your session and pass its name to program DFSRRC00 as a parameter. This member is the sequential file you allocate to ddname FOCPSB.</p>

To invoke the Interface interactively from this environment, first add IMS.RESLIB and 'prefix.FOCLIB.LOAD' to the allocation for ddname STEPLIB in your TSO logon procedure. Execute the following CLIST after editing it to conform to your site's standards

```

PROC 0
CONTROL NOMSG
FREE DDNAME(FOCPSB MASTER FOCEXEC ERRORS USERLIB)
ALLOC F(DFSRESLB) DA('IMS.RESLIB') SHR REUSE
ALLOC F(DFSTROUT) DUMMY REUSE
ALLOC F(IEFRDER) DUMMY REUSE
ALLOC F(DFSVSAMP) DA('IMS.DFSVSAMP(bufpool)') SHR REUSE
ALLOC F(IMS) DA('IMS.DBDLIB' +
                'IMS.PSBLIB') SHR REUSE
ALLOC F(PATDB01) DA('IMS.PATIENT.DB') SHR REUSE
ALLOC F(PATDBIX) DA('IMS.PATIENT.IX') SHR REUSE
ALLOC F(PATDBIX1) DA('IMS.PATIENT.IX1') SHR REUSE
ALLOC F(PATDBIX2) DA('IMS.PATIENT.IX2') SHR REUSE
ALLOC F(PATDBIX3) DA('IMS.PATIENT.IX3') SHR REUSE
ALLOC F(MASTER) DA('prefix.MASTER.DATA') SHR REUSE
ALLOC F(FOCEXEC) DA('prefix.FOCEXEC.DATA') SHR REUSE
ALLOC F(FOCPSB) DA('prefix.FOCPSB.DATA(psbname)') SHR REUSE
ALLOC F(ERRORS) DA('prefix.ERRORS.DATA') SHR REUSE
ALLOC F(USERLIB) DA('prefix.IMS.LOAD' +
                    'prefix.FUSELIB.LOAD') SHR REUSE
CALL 'IMS.RESLIB(DFSRR00)' 'DLI, FOCUS, psbname'

```

where:

bufpool Is the member that contains your VSAM buffer pool information.

prefix Is the high-level qualifier for your site's FOCUS production libraries.

psbname Is the name of the PSB to use.

Note:

- You must allocate the IMS databases, PSBs, and DBDs with this configuration.
The ddnames allocated to the IMS databases are from the DD1 parameters in the relevant IMS DBDs. In the sample, the databases are the PATDB databases described in Appendix A, *Sample File Descriptions*.
- If you specify the parameter DBB instead of DLI on the EXEC card, you must include the ACB library in the allocation for ddname IMS. In addition, you must allocate a second dataset, IMS.ACBLIB, to ddname IMSACB.

Environments

To invoke the Interface as a batch job, submit the following JCL after editing it to conform to your site's standards and adding a JOB card

```
//STEP1 EXEC PGM=DFSRR00,REGION=nn,PARM='DLI, FOCUS, psbname'
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
// DD DSN=IMS.RESLIB,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
// DD DSN=prefix.FUSELIB.LOAD,DISP=SHR
//IMS DD DSN=IMS.DBDLIB,DISP=SHR
// DD DSN=IMS.PSBLIB,DISP=SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//MASTER DD DSN=prefix.MASTER.DATA,DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//FOCPSB DD DSN=prefix.FOCPSB.DATA(psbname),DISP=SHR
//DFSVSAMP DD DSN=IMS.DFSVSAMP(bufpool),DISP=SHR
//IEFRDER DD DUMMY
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//DFSTROUT DD DUMMY
//PATDB01 DD DSN=IMS.PATIENT.DB,DISP=SHR
//PATDBIX DD DSN=IMS.PATIENT.IX,DISP=SHR
//PATDBIX1 DD DSN=IMS.PATIENT.IX1,DISP=SHR
//PATDBIX2 DD DSN=IMS.PATIENT.IX2,DISP=SHR
//PATDBIX3 DD DSN=IMS.PATIENT.IX3,DISP=SHR
//SYSPRINT DD SYSOUT=*
//OFFLINE DD SYSOUT=*
//SYSIN DD *
TABLE FILE ...
.
.
.
END
FIN
/*
//
```

where:

<i>nn</i>	Is the region size.
<i>psbname</i>	Is the name of the PSB to use.
<i>prefix</i>	Is the high-level qualifier for your site's FOCUS production libraries.
<i>bufpool</i>	Is the member that contains your VSAM buffer pool information.

Note:

- You must allocate the IMS databases, PSBs, and DBDs with this configuration.
The ddnames allocated to the IMS databases are from the DD1 parameters in the relevant IMS DBDs. In the sample, the databases are the PATDB databases described in Appendix A, *Sample File Descriptions*.
- If you specify the parameter DBB instead of DLI on the EXEC card, you must include the ACB library in the allocation for ddname IMS. In addition, you must allocate a second dataset, IMS.ACBLIB, to ddname IMSACB.

5.4 Summary Chart

The following chart summarizes the environments available with the Interface:

IMS or CICS Component	DFSRR00 or DFHDRP Loads:	Mode	Advantages	Disadvantages
DBCTL (<i>Access to IMS Through DBCTL</i>)	n/a	n/a	Accesses online databases Provides security No XMI server No communication resources PCB sharing Dynamic PSB selection Works with DEDB databases Works with MSO	No MSDB databases Needs IMS 3.1 or up
DFSRR00 (<i>Initiating the XMI Server in BMP Mode: PARM='BMP,XMI ,psbname'</i>)	XMI server	BMP	Accesses online databases Accesses Fast Path Works with MSO	Needs IMS control region running Needs communication resources Needs duplicate PCBs for multiple users

IMS or CICS Component	DFSRRRC00 or DFHDRP Loads:	Mode	Advantages	Disadvantages
DFSRRRC00 (Section 5.2.2, <i>Initiating the XMI Server in DLI Mode:</i> <i>PARM= 'DLI,XMI,psbname')</i>	XMI server	DLI	No IMS control region Works with MSO	Does not retrieve updated records Needs communication resources Needs duplicate PCBs for multiple users No Fast Path
DFHDRP (Section 5.2.3, <i>Initiating the XMI Server With Databases Under Control of CICS)</i>	XMI server	n/a	Accesses online databases Works with MSO	Needs CICS control region running with DL/I SAS Needs communication resources Needs duplicate PCBs for multiple users No Fast Path Not available with IMS 5.1
DFSRRRC00 (Section 5.3.1, <i>Invoking the Interface in BMP Mode:</i> <i>PARM= 'BMP,FOCUS,psbname')</i>	FOCUS	BMP	Accesses online databases Quicker access than XMI server No communication resources Accesses Fast Path	Does not work with MSO Needs IMS control region running
DFSRRRC00 (Section 5.3.2, <i>Invoking the Interface in DLI Mode:</i> <i>PARM= 'DLI,FOCUS,psbname')</i>	FOCUS	DLI	No IMS control region Quicker access than XMI server No communication resources	Does not work with MSO No Fast Path Does not retrieve updated records

5.5 Environment Switching

You can toggle between the XMI server and DBCTL environments by allocating the appropriate datasets and issuing the relevant SET commands. For more information about SET commands, see Section 5.1.3, *Implementing DBCTL*.

5.5.1 Switching to DBCTL From the XMI Server

If you want to switch from the XMI server environment to the DBCTL environment, you must do the following:

1. Allocate the FOCPSB dataset to ddname FOCPSB in your FOCUS address space; the identifier “prefix” represents the high-level qualifier for your FOCUS production libraries:

From	Allocation
The CLIST that invokes FOCUS	ALLOC F(FOCPSB) DA('prefix.FOCPSB.DATA') SHR REUSE
A FOCUS session	{MVS TSO} ALLOC F(FOCPSB) DA('prefix.FOCPSB.DATA')SHR REU
Batch FOCUS JCL or MSO server JCL	//FOCPSB DD DSN=prefix.FOCPSB.DATA,DISP=SHR
An MSO or FOCUS session	DYNAM ALLOC FILE FOCPSB DATASET prefix.FOCPSB.DATA SHR

2. If you are running TSO, identify the suffix for the DRA Startup Table (if you are running MSO, this setting is included in your configuration file, as described in Appendix D, *Installation Instructions*)

```
{MVS|TSO} IMS SET IMSPZP nn
```

where:

nn Is the suffix for the DRA Startup Table, chosen during installation (see Appendix D, *Installation Instructions*).

3. Turn on DBCTL mode by issuing the following:

```
{MVS|TSO} IMS SET DBCTL ON
```

5.5.2 Switching to the XMI Server From DBCTL

If you want to switch from the DBCTL environment to the XMI server environment, you must do the following:

1. Allocate a communication file to ddname FOCBMP in your FOCUS address space. You can either include this allocation in the CLIST or JCL that invokes FOCUS (or in your MSO server JCL) or you can allocate it dynamically. *The Communication File* in Section 5.2.5, *Additional Instructions for Using the XMI Server*, describes how to allocate a communication file.
2. Turn off DBCTL mode by issuing the following:

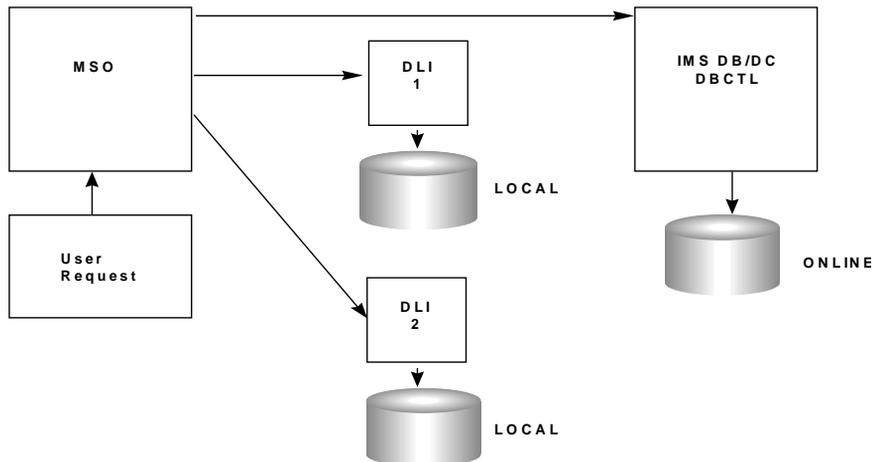
```
{MVS|TSO} IMS SET DBCTL OFF
```

5.5.3 Configuring the IMS Interface to Support Non-stop Processing

The IMS Interface has the ability to read IMS data in several different environments. The current release of the IMS Interface has been enhanced to support non-stop processing. This enhancement gives the Interface the ability to switch the access mode transparently so that reporting from IMS will continue even when one environment is not accessible. DBCTL must be installed to take advantage of this feature.

Although this feature works in a batch environment, it is geared towards an MSO environment.

Below is a diagram that explains how this feature works.



1. The user issues a TABLE or SQL SELECT against an IMS database.
2. The request is passed to IMS via DBCTL which is the default access mode.
3. Sometime during the day, DBCTL access is terminated.
4. The user issues another request.
5. The IMS Interface checks DBCTL availability and determines that DBCTL is not accessible.
6. The request is passed to the first available DLI XMI Server for processing.
7. When DBCTL becomes available, all requests will be passed to it for processing.

This feature can be configured to route all requests to MSO or it can be configured on a user by user basis via the user profile.

To enable this feature, modify your MSO configuration file as shown in Appendix D, *Installation Instructions*.

Note:

- Allocate the required IMS databases to each XMI Server. This is necessary to assure that the users have access to the same data that they had through DBCTL.
- If you wish to force this type of processing on all users that log on to MSO, place the following SET command in the FOCPARM member in your ERRORS PDS:

```
SET IMSNONSTOP=ON
```

- You can also set this processing option on a user-by-user basis by placing the following SET command in the user's profile:

```
TSO IMS SET NONSTOP ON
```

6 Security

Three types of security are available with the IMS/DB Interface:

- FOCUS provides the option of adding security restrictions and passwords to Master Files. The FOCUS DBA (database administrator) can encrypt Master Files to conceal this security information. See the *FOCUS for IBM Mainframe Users Manual* for information on FOCUS DBA security and the ENCRYPT command.
- The DBCTL environment, when accessed through MSO, provides access to standard security systems through the standard SAF interface. With the SAF interface, your site can use security products such as RACF, CA-TOP SECRET, and CA-ACF2 to restrict access to PSBs. Before allowing access to a particular PSB, the security system verifies that the user is authorized to read the PSB.
- The XMI Server includes a security exit that allows a site to secure the environment at the user level. The system administrator can define which PCB(s) within the PSB can be accessed by a given user.

Note:

- Interface users accessing IMS without DBCTL, or in a TSO environment, cannot benefit from the PSB security mechanism implemented in DBCTL because it requires the environment to run authorized, an unacceptable solution to many sites.
- A site can secure PSBs by enabling the IMS/ESA Application Group Name (AGN) feature. Steps for implementing this feature are documented in the Information Builders technical memorandum TM7910, *IMS Security via Application Group Name*.

This chapter discusses DBCTL security and XMI server security.

6.1 DBCTL Security

The object of the security feature is to ensure that users access only those PSBs for which they have authorization. The MSO server can query standard security systems through the standard SAF interface before allowing a user to access a PSB.

The DBCTL function is tested and verified with the RACF product; other SAF products using the identical calls should perform properly when installed and verified by your site's security administrator.

RACF comes with several predefined security classes. Customer sites can use an existing class (such as PCICSPSB) or define a resource class specifically for DBCTL use.

The following example illustrates how to define a PSB resource through a PCICSPSB profile to RACF and how to grant users permission to access the resource.

```
RDEFINE PCICSPSB (psbname) UACC(NONE) NOTIFY(sys_admin_userid)
PERMIT psbname CLASS(PCICSPSB) ID(user_or_group [user_or_group ...])
ACCESS(READ)
```

where:

<i>psbname</i>	Is the PSB name to be protected by RACF.
<i>sys_admin_userid</i>	Is the userid of the system administrator.
<i>user_or_group</i>	Authorizes the userids and/or user groups listed in the PERMIT command to read the specified PSB. Separate items in the list with blanks.

To implement DBCTL security, include the IMSSEC=ON and IMSCLASS attributes in the MSO configuration file during installation (see Appendix D, *Installation Instructions*).

The syntax is

```
IMSSEC = {ON|OFF}
IMSCLASS = classname
```

where:

ON	Activates DBCTL security. This setting requires the MSO server in authorized mode.
<u>OFF</u>	Does not implement DBCTL security. This value is the default.
<i>classname</i>	Is a valid class that contains the security rule. The system administrator can define any class name required by the site's security implementation.

At run time, after the PSB is selected but prior to scheduling it, the MSO server issues a call to the security system and verifies that the user is authorized to read the PSB.

Note: The ON setting for IMSSEC requires the MSO server to be in authorized mode.

6.2 XMI Server Security

In the past the only form of security available for the XMI Server was through RACF, CA-Top Secret, or CA-ACF2. You would apply this security to the FOCBMP dataset, which denied access to the server by not allowing the user to allocate the dataset.

The problem with this type of security was that the user had access to all PCBs within the PSB associated with the XMI Server.

The XMI Server has been enhanced to include a security exit (IMSECHK) that allows a site to secure the environment at the user level. This means that the site's security administrator can define which PCBs within the PSB associated with the server can be accessed by a given user.

6.2.1 How Does the Exit Work

When you issue a TABLE or SQL SELECT against an IMS file, the following things occur:

- The Userid and Master File name are passed to the XMI server.
- The XMI Server tries to load module IMSECHK.

If the IMSECHK module is not found, a message is posted in the XMI server JES MSG LOG.

If the IMSECHK module is found, it is loaded, called, and passed the Userid and Filename parameters. Userid and Filename (Master File) are eight-byte ALPHA values.

Code the exit to perform the appropriate security check:

- If the user is allowed to access the file, the exit should return a zero return code.
- If the user fails the check, the exit should return a non-zero return code. FOCUS will display a FOC4212 when a user fails the security check.

Example: Exit

```

IMSECHK  CSECT
        USING *,R15
        B      ARROUND
        DC     C'*** IMS SECURITY EXIT FOR XMI -- '
        DC     C'&SYSDATE',C' ** '
        DS     0H
ARROUND  EQU   *
        DROP  R15
        STM   R14,R12,12(R13)
        USING IMSECHK,R12
        LR    R12,R15
        LR    R3,R1
        GETMAIN R,LV=LUSAREA
        XC    0(LUSAREA,R1),0(R1)
        ST    R13,4(R1)
        ST    R1,8(R13)
        LR    R13,R1
        USING USAREA,R13
        LM    R4,R5,0(R3)
        MVC   USER(8),0(R4)
        MVC   FILE(8),0(R5)
***** ACTUAL BODY OF THE EXIT *****
        LA    R11,=V(TABLE)      Link to the list of users
        LTR   R11,R11             Is it linked?
        BZ    GOOD               BIF nothing to check
        L     R10,0(R11)         Number of elements in the table
        L     R9,4(R11)         Number of MFDs in the element
        LA    R11,8(R11)        Points to the first one
        USING ELEMENT,R11       Map it
CHECKU   EQU   *
        CLC   USER(8),USERID    Does user registered in the table?
        BE    CHECKF            BIF found
        LA    R11,LELEMENT(R11) Bump to the next element
        BCT   R10,CHECKU        Continue to search
        B     GOOD              User can access any file if NOT found
CHECKF   EQU   *
        LA    R8,MFDS           Address of the MFD's list
COMPMFD  EQU   *
        CLC   FILE(8),0(R8)     Compare names
        BE    GOOD              BIF found
        LA    R8,8(R8)         Bump to the next MFD
        BCT   R9,COMPMFD
        LA    R5,1             FAILED if not found in the list
        B     EXIT
GOOD     EQU   *
        LA    R5,0             PASSED

```

```

*****      EXIT FROM THE EXIT *****
EXIT      EQU      *
          DROP    R13

          LR      R1,R13
          L       R13,4(R13)
          FREEMAIN R,LV=LUSAREA,A=(R1)
          LR      R15,R5      RETURN RC TO THE CALLER
          L       R14,12(R13)
          LM      R0,R12,20(R13)
          BR      R14

USAREA    DSECT
SAVE      DS      18A
USER      DS      CL8
FILE      DS      CL8
LUSAREA   EQU      *-USAREA
ELEMENT   DSECT
USERID    DS      CL8
MFDS      DS      10CL8
LELEMENT  EQU      *-ELEMENT
          REGEQU

*****
* Table of users and accessible MFDS *
* Separate Control Section          *
*****
TABLE     CSECT
ENTRIES   DC      A(TABLEL/ENTRYL)
MFDS#     DC      A(10)
ENTRY     EQU      *
USER1     DC      CL8'PGMJVP5'
          DC      CL8'*****',9CL8' '
ENTRYL    EQU      *-ENTRY
USER2     DC      CL8'PGMBOP5'
          DC      CL8'*****',9CL8' '
USER3     DC      CL8'PMSMJB5'
          DC      CL8'*****',9CL8' '
USER4     DC      CL8'PMSHEB'
          DC      CL8'PATINFO',9CL8' '
USER5     DC      CL8'EDANTK5'
          DC      CL8'*****',9CL8' '
TABLEL    EQU      *-ENTRY
          END

```

6.2.2 Installation of IMSECHK Exit

To install the exit you must do the following:

- Create an exit that accepts two parameters: Userid (8-byte ALPHA) and Filename (8-byte ALPHA).
- Assemble and link your exit.
- Concatenate the resultant load module in the STEPLIB of the XMI server JCL.

The exit will be called each time the user makes a request against the XMI server.

6.2.3 Tracing the Exit

To trace the exit, you can allocate ddname FSTRACE2 in the XMI server JCL. The trace shows the Userid, Filename and return code used and returned by the exit. The following messages will appear in the trace output.

```
IMSRV: Security Check Exit is called.  User:  %8s;  FILE:  %8s
```

```
IMSRV: Security Check Exit returned %d, rc
```

The syntax to allocate ddname FSTRACE2 is as follows:

```
//FSTRACE2 DD SYSOUT=*,DCB=BLKSIZE=80
```

A Sample File Descriptions

This appendix contains Master Files, FOCPSBs, PSBs, and DBDs for sample IMS databases cited in previous chapters. The databases documented in this appendix are shown in the following table:

Database	Description
DI21PART	Sample HDAM database included by IBM with IMS.
PATDB01	Sample HIDAM database (not shipped to user sites); demonstrates secondary indexes.
EMPDB01	Sample HIDAM database (not shipped to user sites); demonstrates partitioning/concatenation of PSBs.
AIHDAM	Sample HDAM database (not shipped to user sites); demonstrates a secondary index on the root key.

Note: FOCPSBs created in prior releases may consist of fixed format records with no attribute keywords. While this earlier format is supported, the recommended format is the comma-delimited format used in the samples in this appendix. See Appendix C, *Release Dependent Interface Features*, for a description of fixed format FOCPSBs.

A.1 DI21PART Sample

IBM ships the DI21PART database, which is an HDAM database, with IMS.

A.1.1 DI21PART DBD

```
PRINT NOGEN
DBD    NAME=DI21PART, ACCESS=(HDAM, VSAM), RMNAME=(DFSHDC40, 4, 80, 500)
DATASET DD1=DI21PART, DEVICE=3380
SEGMENT NAME=PARTROOT, PARENT=0, BYTES=50, FREQ=250
FIELD  NAME=(PARTKEY, SEQ), TYPE=C, BYTES=17, START=1
SEGMENT NAME=STANINFO, PARENT=PARTROOT, BYTES=85, FREQ=1
FIELD  NAME=(STANKEY, SEQ), TYPE=C, BYTES=2, START=1
SEGMENT NAME=STOKSTAT, PARENT=PARTROOT, BYTES=160, FREQ=2
FIELD  NAME=(STOCKEY, SEQ), TYPE=C, BYTES=16, START=1
SEGMENT NAME=CYCCOUNT, PARENT=STOKSTAT, BYTES=25, FREQ=1
FIELD  NAME=(CYCLKEY, SEQ), TYPE=C, BYTES=2, START=1
SEGMENT NAME=BACKORDR, PARENT=STOKSTAT, BYTES=75, FREQ=0
FIELD  NAME=(BACKKEY, SEQ), TYPE=C, BYTES=10, START=1
DBDGEN
FINISH
END
```

A.1.2 PSB to Access DI21PART

```

PCB          TYPE=TP,MODIFY=YES,EXPRESS=YES
PCB          TYPE=TP,EXPRESS=NO,MODIFY=YES,SAMETRM=YES
PCB          TYPE=DB,DBDNAME=DI21PART,PROCOPT=GO,KEYLEN=43
SENSESEG    NAME=PARTROOT,PARENT=0
SENSESEG    NAME=STANINFO,PARENT=PARTROOT
SENSESEG    NAME=STOKSTAT,PARENT=PARTROOT
SENSESEG    NAME=CYCCOUNT,PARENT=STOKSTAT
SENSESEG    NAME=BACKORDR,PARENT=STOKSTAT
PSBGEN     LANG=COBOL,PSBNAME=FOCSD,CMPAT=YES
END
    
```

This PSB is member FOCSD in the PSB dataset because PSBNAME=FOCSD.

A.1.3 FOCPSB to Access DI21PART

```

FOCPSB=EXTENDED,$
PCBNAME=    ,PCBTYPE=TERM,$
PCBNAME=    ,PCBTYPE=TERM,$
PCBNAME=    ,PCBTYPE=TERM,$
PCBNAME=DI21PART,PCBTYPE=DB,$
    
```

This FOCPSB is member FOCSD in the FOCPSB dataset because it must have the same member name as its corresponding PSB.

A.1.4 DI21PART Master File

```

FILE=DI21PART    ,SUFFIX=IMS,$
SEGNAME=PARTROOT ,PARENT=,SEGTYPE=S2,$
  FIELD=PARTKEY  ,ALIAS=PARTKEY.HKY ,USAGE=A17 ,ACTUAL=A17 ,,$
  FIELD=SKIP1    ,ALIAS=SKIP1      ,USAGE=A33 ,ACTUAL=A33 ,,$
SEGNAME=STANINFO ,PARENT=PARTROOT,SEGTYPE=S2,$
  FIELD=STANKEY  ,ALIAS=STANKEY.KEY ,USAGE=A2   ,ACTUAL=A2   ,,$
  FIELD=SKIP2    ,ALIAS=SKIP2      ,USAGE=A83 ,ACTUAL=A83 ,,$
SEGNAME=STOKSTAT ,PARENT=PARTROOT,SEGTYPE=S2,$
  FIELD=STOCKEY  ,ALIAS=STOCKEY.KEY ,USAGE=A16 ,ACTUAL=A16 ,,$
  FIELD=SKIP3    ,ALIAS=,SKIP3      ,USAGE=A124,ACTUAL=A124 ,,$
SEGNAME=CYCCOUNT ,PARENT=STOKSTAT,SEGTYPE=S2,$
  FIELD=CYCCKEY  ,ALIAS=CYCCKEY.KEY ,USAGE=A2   ,ACTUAL=A2   ,,$
  FIELD=SKIP4    ,ALIAS=SKIP4      ,USAGE=A23 ,ACTUAL=A23 ,,$
SEGNAME=BACKORDR ,PARENT=STOKSTAT,SEGTYPE=S2,$
  FIELD=BACKEY   ,ALIAS=BACKEY.KEY   ,USAGE=A10 ,ACTUAL=A10 ,,$
  FIELD=SKIP5    ,ALIAS=SKIP5      ,USAGE=A65 ,ACTUAL=A65 ,,$
    
```

This Master File is member DI21PART in the Master File dataset because its corresponding PCB in the DI21PART FOCPSB specifies PCBNAME=DI21PART.

A.1.5 DI21PART Access File

```
PSB=FOCSD , $
```

This Access File is member DI21PART in the Access File dataset because it must have the same member name as its corresponding Master File. It can be used only in the DBCTL environment.

A.2 PATDB01 Sample

The PATDB01 database has a primary index because it is a HIDAM database. It is also defined with three secondary indexes. Chapter 3, *Creating FOCUS Descriptions*, describes how to create a Master File to use secondary indexes.

This section illustrates the following:

- DBDs for the database and the indexes.
- A PSB with four PCBs: one that uses the primary index and three others that use a secondary index as the main entry point to the database.
- The FOCPSB that corresponds to the PSB.
- A Master File, PATINFO, that describes the primary and three secondary indexes.

A.2.1 PATDB01 DBD

Five DBDs are associated with PATDB01: the DBD for the database, the DBD for the primary index, and one DBD for each of the three secondary indexes.

Database DBD for PATDB01

PATDB01 DBD:

```

PRINT NOGEN
DBD      NAME=PATDB01,ACCESS=(HIDAM,VSAM)
DATASET  DD1=PATDB01,DEVICE=3380,BLOCK=4096,SCAN=5
*
SEGM NAME=PATINFO,PTR=H,PARENT=0,BYTES=233
  FIELD NAME=(SSN,SEQ,U),BYTES=9,START=26,TYPE=C
  LCHILD NAME=(SEGIX,PATDBIX),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=6,START=1,TYPE=C
  FIELD NAME=REVSEQ,BYTES=6,START=7,TYPE=C
  FIELD NAME=SSNALPHA,BYTES=9,START=35,TYPE=C
  FIELD NAME=EMPID,BYTES=12,START=44,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=56,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=68,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=89,TYPE=C
  FIELD NAME=PATID,BYTES=10,START=176,TYPE=C
  FIELD NAME=/SX1
*
  LCHILD NAME=(SEGIX1,PATDBIX1),PTR=INDX
  XDFLD  NAME=IXNAME,SRCH=(LNAME,FNAME),
        SUBSEQ=/SX1,NULLVAL=BLANK
        X
*
  LCHILD NAME=(SEGIX2,PATDBIX2),PTR=INDX
  XDFLD  NAME=IXCOMP,SRCH=(SSNALPHA,EMPID,LNAME)
*
  LCHILD NAME=(SEGIX3,PATDBIX3),PTR=INDX
  XDFLD  NAME=IXADMD,SRCH=(ADMDATE),
        SUBSEQ=/SX1,NULLVAL=BLANK
        X
*
DBDGEN
FINISH
END

```

Primary Index DBD for PATDB01

PATDBIX DBD:

```

PRINT NOGEN
DBD      NAME=PATDBIX,ACCESS=INDEX
DATASET  DD1=PATDBIX,DEVICE=3380
*
SEGM NAME=SEGIX,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1,TYPE=C
  LCHILD NAME=(PATINFO,PATDB01),INDEX=SSN
DBDGEN
FINISH
END

```

Secondary Index DBDs for PATDB01

PATDBIX1 DBD:

```

PRINT NOGEN
DBD      NAME=PATDBIX1,ACCESS=INDEX
DATASET  DD1=PATDBIX1,DEVICE=3380
*
SEGM NAME=SEGIX1,PARENT=0,BYTES=28
  FIELD NAME=(IXNAMEIX,SEQ,U),BYTES=28,START=1
  LCHILD NAME=(PATINFO,PATDB01),INDEX=IXNAME,PTR=SNGL
DBDGEN
FINISH
END

```

PATDBIX2 DBD:

```

PRINT NOGEN
DBD      NAME=PATDBIX2,ACCESS=INDEX
DATASET  DD1=PATDBIX2,DEVICE=3380
*
SEGM NAME=SEGIX2,PARENT=0,BYTES=33
  FIELD NAME=(IXCOMPIX,SEQ,U),BYTES=33,START=1
  LCHILD NAME=(PATINFO,PATDB01),INDEX=IXCOMP,PTR=SNGL
DBDGEN
FINISH
END

```

PATDBIX3 DBD:

```

PRINT NOGEN
DBD      NAME=PATDBIX3,ACCESS=INDEX
DATASET  DD1=PATDBIX3,DEVICE=3380
*
SEGM NAME=SEGIX3,PARENT=0,BYTES=12
  FIELD NAME=(IXADMIX,SEQ,U),BYTES=12,START=1
  LCHILD NAME=(PATINFO,PATDB01),INDEX=IXADM,PTR=SNGL
DBDGEN
FINISH
END

```

A.2.2 PSB to Access PATDB01

```
PCB          TYPE=TP , MODIFY=YES , EXPRESS=YES
PCB          TYPE=TP , EXPRESS=NO , MODIFY=YES , SAMETRM=YES
*
PCB          TYPE=DB , DBDNAME=PATDB01 , PROCOPT=GO , KEYLEN=9
SENSESEG    NAME=PATINFO , PARENT=0
*
PCB          TYPE=DB , DBDNAME=PATDB01 , PROCOPT=GO , KEYLEN=28 , PROCSEQ=PATDBIX1
SENSESEG    NAME=PATINFO , PARENT=0
*
PCB          TYPE=DB , DBDNAME=PATDB01 , PROCOPT=GO , KEYLEN=33 , PROCSEQ=PATDBIX2
SENSESEG    NAME=PATINFO , PARENT=0
*
PCB          TYPE=DB , DBDNAME=PATDB01 , PROCOPT=GO , KEYLEN=12 , PROCSEQ=PATDBIX3
SENSESEG    NAME=PATINFO , PARENT=0
*
PSBGEN      LANG=COBOL , PSBNAME=TSTPSB01 , CMPAT=YES
END
```

This PSB is member TSTPSB01 in the PSB dataset because PSBNAME=TSTPSB01.

A.2.3 FOCPSB to Access PATDB01

```
FOCPSB=EXTENDED , $
PCBNAME=    , PCBTYP=TERM , $
PCBNAME=    , PCBTYP=TERM , $
PCBNAME=    , PCBTYP=TERM , $
PCBNAME=PATINFO , PCBTYP=DB , $
PCBNAME=IXNAME , PCBTYP=DB , $
PCBNAME=IXCOMP , PCBTYP=DB , $
PCBNAME=IXADMD , PCBTYP=DB , $
```

This FOCPSB is member TSTPSB01 in the FOCPSB dataset because it must have the same member name as its corresponding PSB. It has four database PCB entries. The first is for the normal entry point into the database and identifies the name of the Master File. Each of the other entries identifies one of the three secondary indexes.

A.2.4 Master File to Access PATDB01

```

FILE=PATDB01 , SUFFIX=IMS , $
SEGNAME=PATINFO , PARENT= , SEGTYPE=S1 , $
FIELD=SEQFIELD , ALIAS=SEQFIELD . IMS , USAGE=A06 , ACTUAL=A06 , $
FIELD=REVSEQ , ALIAS=REVSEQ . IMS , USAGE=A06 , ACTUAL=A06 , $
FIELD=SALARY , ALIAS= , USAGE=A08 , ACTUAL=A08 , $
FIELD=OT_HR_PAY , ALIAS= , USAGE=A05 , ACTUAL=A05 , $
FIELD=SSN , ALIAS=SSN . KEY , USAGE=A09 , ACTUAL=A09 , $
FIELD=SSNALPHA , ALIAS=SSNALPHA . IMS , USAGE=A09 , ACTUAL=A09 , $
FIELD=EMPLOYEEID , ALIAS=EMPID . IMS , USAGE=A12 , ACTUAL=A12 , $
FIELD=LAST_NAME , ALIAS=LNAME . IMS , USAGE=A12 , ACTUAL=A12 , $
FIELD=FIRST_NAME , ALIAS=FNAME . IMS , USAGE=A12 , ACTUAL=A12 , $
FIELD=DATE_OF_BIRTH , ALIAS= , USAGE=A08 , ACTUAL=A08 , $
FIELD=RACE , ALIAS= , USAGE=A01 , ACTUAL=A01 , $
FIELD=ADMIT_DATE , ALIAS=ADMDATE . IMS , USAGE=A08 , ACTUAL=A08 , $
FIELD=ADMIT_TYPE , ALIAS= , USAGE=A01 , ACTUAL=A01 , $
FIELD=DISPOSITION , ALIAS= , USAGE=A02 , ACTUAL=A02 , $
FIELD=TRANSFER_DATE , ALIAS= , USAGE=A08 , ACTUAL=A08 , $
FIELD=ALLERGY1 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
FIELD=ALLERGY2 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
FIELD=ALLERGY3 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
FIELD=ALLERGY4 , ALIAS= , USAGE=A15 , ACTUAL=A15 , $
FIELD=HOUSING , ALIAS= , USAGE=A03 , ACTUAL=A03 , $
FIELD=RPR , ALIAS= , USAGE=A01 , ACTUAL=A01 , $
FIELD=URIN , ALIAS= , USAGE=A01 , ACTUAL=A01 , $
FIELD=PRACTITIONAR , ALIAS= , USAGE=A02 , ACTUAL=A02 , $
FIELD=SHIFT , ALIAS= , USAGE=A01 , ACTUAL=A01 , $
FIELD=PATINET_ID , ALIAS=PATID . IMS , USAGE=A12 , ACTUAL=A12 , $
FIELD=WHO_ADDED , ALIAS= , USAGE=A10 , ACTUAL=A10 , $
FIELD=DATE_ADDED , ALIAS= , USAGE=A08 , ACTUAL=A08 , $
FIELD=WHO_EDITED , ALIAS= , USAGE=A10 , ACTUAL=A10 , $
FIELD=DATE_EDITED , ALIAS= , USAGE=A08 , ACTUAL=A08 , $
FIELD=STATION_ID , ALIAS= , USAGE=A12 , ACTUAL=A12 , $
FIELD=DIABETIC , ALIAS= , USAGE=A01 , ACTUAL=A01 , $
FIELD=DIALYSIS , ALIAS= , USAGE=A01 , ACTUAL=A01 , $

GROUP=NAMEIX , ALIAS=IXNAME . SKY , , USAGE=A24 , , ACTUAL=A24 , $
FIELD=NAMEL , ALIAS=LAST_NAME , , USAGE=A12 , , ACTUAL=A12 , $
FIELD=NAMEF , ALIAS=FIRST_NAME , , USAGE=A12 , , ACTUAL=A12 , $

GROUP=COMPIX , ALIAS=IXCOMP . SKY , , USAGE=A33 , , ACTUAL=A33 , $
FIELD=SSN_ALPHA , ALIAS=SSNALPHA , , USAGE=A09 , , ACTUAL=A09 , $
FIELD=EMPLOYEE_ID , ALIAS=EMPLOYEEID , , USAGE=A12 , , ACTUAL=A12 , $
FIELD=LASTNAME , ALIAS=LAST_NAME , , USAGE=A12 , , ACTUAL=A12 , $

GROUP=ADMDIX , ALIAS=IXADMD . SKY , , USAGE=A08 , , ACTUAL=A08 , $
FIELD=SSN_ALPHA , ALIAS=SSNALPHA , , USAGE=A08 , , ACTUAL=A08 , $

```

This Master File is member PATINFO in the Master File dataset because the first database PCB in the TSTPSB01 FOCPSB specifies PCBNAME=PATINFO.

A.3 EMPDB Sample

EMPDB01, EMPDB02, and EMPDB03 are HIDAM databases that illustrate partitioning and concatenation of PCBs in the FOCPSB (see Chapter 3, *Creating FOCUS Descriptions*).

This section illustrates the following:

- The EMPDB01, EMPDB02, and EMPDB03 DBDs for the three partitions of the database and, since these are HIDAM databases, the EMPDBIX1, EMPDBIX2, and EMPDBIX3 DBDs for the index databases.
- A PSB with three PCBs, each corresponding to one partition of the database.
- The FOCPSB corresponding to the PSB. It includes the LOWVALUE and HIGHVALUE attributes for each PCB and the CONCATNAME record to define the concatenation.
- The EMPDB01, EMPDB02, and EMPDB03 Master Files for the individual partitions and the EMPDBJ Master File for the concatenation.

A.3.1 EMPDB DBDs

Each of the following sections describes one partition of the EMPDB database.

EMPDB01 DBD

Database DBD:

```

PRINT NOGEN
DBD      NAME=EMPDB01,ACCESS=(HIDAM,VSAM)
DATASET DD1=EMPDB01,DEVICE=3380,BLOCK=4096,SCAN=5
*
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX1,EMPDBIX1),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
*
DBDGEN
FINISH
END

```

Index DBD:

```

PRINT NOGEN
DBD      NAME=EMPDBIX1,ACCESS=INDEX
DATASET DD1=EMPDBIX1,DEVICE=3380
*
SEGM NAME=SEGIX1,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB01),INDEX=SSNALPHA
DBDGEN
FINISH
END

```

EMPDB02 DBD

Database DBD:

```

PRINT NOGEN
DBD      NAME=EMPDB02,ACCESS=(HIDAM,VSAM)
DATASET DD1=EMPDB02,DEVICE=3380,BLOCK=4096,SCAN=5
*
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX2,EMPDBIX2),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
*
DBDGEN
FINISH
END

```

Index DBD:

```

PRINT NOGEN
DBD      NAME=EMPDBIX2,ACCESS=INDEX
DATASET DD1=EMPDBIX2,DEVICE=3380
*
SEGM NAME=SEGIX2,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB02),INDEX=SSNALPHA
DBDGEN
FINISH
END

```

EMPDB03 DBD

Database DBD:

```
PRINT NOGEN
DBD      NAME=EMPDB03,ACCESS=(HIDAM,VSAM)
DATASET  DD1=EMPDB03,DEVICE=3380,BLOCK=4096,SCAN=5
*
SEGM NAME=EMPINFO,PTR=H,PARENT=0,BYTES=212
  FIELD NAME=(SSNALPHA,SEQ,U),BYTES=9,START=18,TYPE=C
  LCHILD NAME=(SEGIX3,EMPDBIX3),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=2,START=1,TYPE=P
  FIELD NAME=REVSEQ,BYTES=6,START=3,TYPE=C
  FIELD NAME=SSN,BYTES=3,START=15,TYPE=P
  FIELD NAME=EMPID,BYTES=12,START=27,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=39,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=51,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=72,TYPE=C
*
DBDGEN
FINISH
END
```

Index DBD:

```
PRINT NOGEN
DBD      NAME=EMPDBIX3,ACCESS=INDEX
DATASET  DD1=EMPDBIX3,DEVICE=3380
*
SEGM NAME=SEGIX3,PARENT=0,BYTES=9
  FIELD NAME=(SSNIX,SEQ,U),BYTES=9,START=1
  LCHILD NAME=(EMPINFO,EMPDB03),INDEX=SSNALPHA
DBDGEN
FINISH
END
```

A.3.2 PSB to Access the EMPDB Databases

```

PCB          TYPE=TP ,MODIFY=YES ,EXPRESS=YES
PCB          TYPE=TP ,EXPRESS=NO ,MODIFY=YES ,SAMETRM=YES
PCB          TYPE=DB ,DBDNAME=EMPDB01 ,PROCOPT=GO ,KEYLEN=9
SENSEG      NAME=EMPINFO ,PARENT=0
PCB          TYPE=DB ,DBDNAME=EMPDB02 ,PROCOPT=GO ,KEYLEN=9
SENSEG      NAME=EMPINFO ,PARENT=0
PCB          TYPE=DB ,DBDNAME=EMPDB03 ,PROCOPT=GO ,KEYLEN=9
SENSEG      NAME=EMPINFO ,PARENT=0
PSBGEN LANG=COBOL ,PSBNAME=EMPPSBJ ,CMPAT=YES
END

```

This PSB is member EMPPSBJ in the PSB dataset because PSBNAME=EMPPSBJ.

A.3.3 FOCPSB to Access the EMPDB Databases

```

FOCPSB=EXTENDED , $
PCBNAME=    , PCBTYP=TERM , $
PCBNAME=    , PCBTYP=TERM , $
PCBNAME=    , PCBTYP=TERM , $
PCBNAME=EMPDB01 , PCBTYP=DB , LOWVALUE=000000001 , HIGHVALUE=000001667 , $
PCBNAME=EMPDB02 , PCBTYP=DB , LOWVALUE=000001668 , HIGHVALUE=000003334 , $
PCBNAME=EMPDB03 , PCBTYP=DB , LOWVALUE=000003335 , HIGHVALUE=000005000 , $
CONCATNAME=EMPDBJ , USE=EMPDB01/EMPDB02/EMPDB03 , $

```

This FOCPSB is member EMPPSBJ in the FOCPSB dataset because it must have the same member name as its corresponding PSB.

A.3.4 Master Files to Access the EMPDB Databases

The EMPDB01, EMPDB02, and EMPDB03 Master Files each correspond to a PCB for one individual partition of the database; the EMPDBJ Master File corresponds to the concatenation of the three partitions.

EMPDB01 Master File

```

FILE=EMPDB01,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD,ALIAS=SEQFIELD.IMS,USAGE=P6,ACTUAL=P2,$
FIELD=REVSEQ,ALIAS=REVSEQ.IMS,USAGE=A6,ACTUAL=A6,$
FIELD=SALARY,ALIAS=,USAGE=P8,ACTUAL=P4,$
FIELD=OT_HR_PAY,ALIAS=,USAGE=P5,ACTUAL=P2,$
FIELD=SSN,ALIAS=SSN.IMS,USAGE=P9,ACTUAL=P3,$
FIELD=SSNALPHA,ALIAS=SSNALPHA.KEY,USAGE=A9,ACTUAL=A9,$
FIELD=EMPID,ALIAS=EMPID.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=ELAST_NAME,ALIAS=ELNAME.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=EFIRST_NAME,ALIAS=EFNAME.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=RACE,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=ADMIT_DATE,ALIAS=ADMDATE.IMS,USAGE=A08,ACTUAL=A08,$
FIELD=ADMIT_TYPE,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=DISPOSITION,ALIAS=,USAGE=A02,ACTUAL=A02,$
FIELD=TRANSFER_DATE,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=ALLERGY1,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY2,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY3,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY4,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=HOUSING,ALIAS=,USAGE=A03,ACTUAL=A03,$
FIELD=RPR,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=URLN,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=PRACTITIONAR,ALIAS=,USAGE=A02,ACTUAL=A02,$
FIELD=SHIFT,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=PATINET_ID,ALIAS=,USAGE=P12,ACTUAL=P4,$
FIELD=WHO_ADDED,ALIAS=,USAGE=A10,ACTUAL=A10,$
FIELD=DATE_ADDED,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=WHO_EDITED,ALIAS=,USAGE=A10,ACTUAL=A10,$
FIELD=DATE_EDITED,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=STATION_ID,ALIAS=,USAGE=A12,ACTUAL=A12,$
FIELD=DIABETIC,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=DIALYSIS,ALIAS=,USAGE=A01,ACTUAL=A01,$

```

This Master File is member EMPDB01 in the Master File dataset because the first database PCB in the EMPDBJ FOCPSB specifies PCBNAME=EMPDB01.

EMPDB02 Master File

```

FILE=EMPDB02,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD,ALIAS=SEQFIELD.IMS,USAGE=P6,ACTUAL=P2,$
FIELD=REVSEQ,ALIAS=REVSEQ.IMS,USAGE=A6,ACTUAL=A6,$
FIELD=SALARY,ALIAS=,USAGE=P8,ACTUAL=P4,$
FIELD=OT_HR_PAY,ALIAS=,USAGE=P5,ACTUAL=P2,$
FIELD=SSN,ALIAS=SSN.IMS,USAGE=P9,ACTUAL=P3,$
FIELD=SSNALPHA,ALIAS=SSNALPHA.KEY,USAGE=A9,ACTUAL=A9,$
FIELD=EMPID,ALIAS=EMPID.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=ELAST_NAME,ALIAS=ELNAME.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=EFIRST_NAME,ALIAS=EFNAME.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=RACE,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=ADMIT_DATE,ALIAS=ADMDATE.IMS,USAGE=A08,ACTUAL=A08,$
FIELD=ADMIT_TYPE,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=DISPOSITION,ALIAS=,USAGE=A02,ACTUAL=A02,$
FIELD=TRANSFER_DATE,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=ALLERGY1,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY2,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY3,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY4,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=HOUSING,ALIAS=,USAGE=A03,ACTUAL=A03,$
FIELD=RPR,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=URIN,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=PRACTITIONAR,ALIAS=,USAGE=A02,ACTUAL=A02,$
FIELD=SHIFT,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=PATINET_ID,ALIAS=,USAGE=P12,ACTUAL=P4,$
FIELD=WHO_ADDED,ALIAS=,USAGE=A10,ACTUAL=A10,$
FIELD=DATE_ADDED,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=WHO_EDITED,ALIAS=,USAGE=A10,ACTUAL=A10,$
FIELD=DATE_EDITED,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=STATION_ID,ALIAS=,USAGE=A12,ACTUAL=A12,$
FIELD=DIABETIC,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=DIALYSIS,ALIAS=,USAGE=A01,ACTUAL=A01,$

```

This Master File is member EMPDB02 in the Master File dataset because the second database PCB in the EMPDSBJ FOCPSB specifies PCBNAME=EMPDB02.

EMPDB03 Master File

```

FILE=EMPDB03,SUFFIX=IMS,$
SEGNAME=EMPINFO,PARENT=,SEGTYPE=S1,$
FIELD=SEQFIELD,ALIAS=SEQFIELD.IMS,USAGE=P6,ACTUAL=P2,$
FIELD=REVSEQ,ALIAS=REVSEQ.IMS,USAGE=A6,ACTUAL=A6,$
FIELD=SALARY,ALIAS=,USAGE=P8,ACTUAL=P4,$
FIELD=OT_HR_PAY,ALIAS=,USAGE=P5,ACTUAL=P2,$
FIELD=SSN,ALIAS=SSN.IMS,USAGE=P9,ACTUAL=P3,$
FIELD=SSNALPHA,ALIAS=SSNALPHA.KEY,USAGE=A9,ACTUAL=A9,$
FIELD=EMPID,ALIAS=EMPID.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=ELAST_NAME,ALIAS=ELNAME.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=EFIRST_NAME,ALIAS=EFNAME.IMS,USAGE=A12,ACTUAL=A12,$
FIELD=DATE_OF_BIRTH,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=RACE,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=ADMIT_DATE,ALIAS=ADMDATE.IMS,USAGE=A08,ACTUAL=A08,$
FIELD=ADMIT_TYPE,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=DISPOSITION,ALIAS=,USAGE=A02,ACTUAL=A02,$
FIELD=TRANSFER_DATE,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=ALLERGY1,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY2,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY3,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=ALLERGY4,ALIAS=,USAGE=A15,ACTUAL=A15,$
FIELD=HOUSING,ALIAS=,USAGE=A03,ACTUAL=A03,$
FIELD=RPR,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=URIN,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=PRACTITIONAR,ALIAS=,USAGE=A02,ACTUAL=A02,$
FIELD=SHIFT,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=PATINET_ID,ALIAS=,USAGE=P12,ACTUAL=P4,$
FIELD=WHO_ADDED,ALIAS=,USAGE=A10,ACTUAL=A10,$
FIELD=DATE_ADDED,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=WHO_EDITED,ALIAS=,USAGE=A10,ACTUAL=A10,$
FIELD=DATE_EDITED,ALIAS=,USAGE=A08,ACTUAL=A08,$
FIELD=STATION_ID,ALIAS=,USAGE=A12,ACTUAL=A12,$
FIELD=DIABETIC,ALIAS=,USAGE=A01,ACTUAL=A01,$
FIELD=DIALYSIS,ALIAS=,USAGE=A01,ACTUAL=A01,$

```

This Master File is member EMPDB03 in the Master File dataset because the third database PCB in the EMPDSBJ FOCPSB specifies PCBNAME=EMPDB03.

EMPDBJ Master File

```

FILE=EMPDBJ , SUFFIX=IMS , $
SEGNAME=EMPINFO , PARENT= , SEGTYPE=S1 , $
FIELD=SEQFIELD      , ALIAS=SEQFIELD . IMS      , USAGE=P6      , ACTUAL=P2 , $
FIELD=REVSEQ        , ALIAS=REVSEQ . IMS      , USAGE=A6      , ACTUAL=A6 , $
FIELD=SALARY        , ALIAS=                      , USAGE=P8      , ACTUAL=P4 , $
FIELD=OT_HR_PAY     , ALIAS=                      , USAGE=P5      , ACTUAL=P2 , $
FIELD=SSN           , ALIAS=SSN . IMS          , USAGE=P9      , ACTUAL=P3 , $
FIELD=SSNALPHA     , ALIAS=SSNALPHA . KEY    , USAGE=A9      , ACTUAL=A9 , $
FIELD=EMPID        , ALIAS=EMPID . IMS       , USAGE=A12     , ACTUAL=A12 , $
FIELD=ELAST_NAME   , ALIAS=ELNAME . IMS     , USAGE=A12     , ACTUAL=A12 , $
FIELD=EFIRST_NAME  , ALIAS=EFNAME . IMS     , USAGE=A12     , ACTUAL=A12 , $
FIELD=DATE_OF_BIRTH , ALIAS=                  , USAGE=A08     , ACTUAL=A08 , $
FIELD=RACE         , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $
FIELD=ADMIT_DATE   , ALIAS=ADMDATE . IMS    , USAGE=A08     , ACTUAL=A08 , $
FIELD=ADMIT_TYPE   , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $
FIELD=DISPOSITION  , ALIAS=                  , USAGE=A02     , ACTUAL=A02 , $
FIELD=TRANSFER_DATE , ALIAS=                  , USAGE=A08     , ACTUAL=A08 , $
FIELD=ALLERGY1     , ALIAS=                  , USAGE=A15     , ACTUAL=A15 , $
FIELD=ALLERGY2     , ALIAS=                  , USAGE=A15     , ACTUAL=A15 , $
FIELD=ALLERGY3     , ALIAS=                  , USAGE=A15     , ACTUAL=A15 , $
FIELD=ALLERGY4     , ALIAS=                  , USAGE=A15     , ACTUAL=A15 , $
FIELD=HOUSING      , ALIAS=                  , USAGE=A03     , ACTUAL=A03 , $
FIELD=RPR          , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $
FIELD=URIN         , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $
FIELD=PRACTITIONAR , ALIAS=                  , USAGE=A02     , ACTUAL=A02 , $
FIELD=SHIFT        , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $
FIELD=PATINET_ID   , ALIAS=                  , USAGE=P12     , ACTUAL=P4 , $
FIELD=WHO_ADDED    , ALIAS=                  , USAGE=A10     , ACTUAL=A10 , $
FIELD=DATE_ADDED   , ALIAS=                  , USAGE=A08     , ACTUAL=A08 , $
FIELD=WHO_EDITED   , ALIAS=                  , USAGE=A10     , ACTUAL=A10 , $
FIELD=DATE_EDITED  , ALIAS=                  , USAGE=A08     , ACTUAL=A08 , $
FIELD=STATION_ID   , ALIAS=                  , USAGE=A12     , ACTUAL=A12 , $
FIELD=DIABETIC     , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $
FIELD=DIALYSIS     , ALIAS=                  , USAGE=A01     , ACTUAL=A01 , $

```

This Master File is member EMPDBJ in the Master File dataset because of the CONCATNAME=EMPDBJ attribute in the EMPPSBJ FOCPSB.

A.4 AIHDAM Sample

AIHDAM is an HDAM database with a secondary index on the root key. Chapter 4, *Reporting Efficiencies*, explains how the index affects report optimization.

A.4.1 AIHDAM DBD

Database DBD:

```

PRINT NOGEN
DBD    NAME=AIHDAM, ACCESS=( HDAM, VSAM ), RMNAME=( DFSHDC40, 4, 80, 500 )
DATASET DD1=AIHDAM, DEVICE=3390, BLOCK=4096, SCAN=5
*
SEGM  NAME=LANGUAGE, PTR=H, PARENT=0, BYTES=19
FIELD NAME=( EMPL6, SEQ, M ), BYTES=4, START=1, TYPE=C
FIELD NAME=LANG6, BYTES=15, START=5, TYPE=C
FIELD NAME=/SX1
*
LCHILD NAME=( SEG6IX1, AIHDAMX ), PTR=INDX
XDFLD  NAME=IXEMP6, SRCH=( EMPL6 ), SUBSEQ=/SX1, NULLVAL=BLANK
*
DBDGEN
FINISH
END

```

Secondary index DBD:

```

PRINT NOGEN
DBD    NAME=AIHDAMX, ACCESS=INDEX
DATASET DD1=AIHDAMX, OVFLW=AIHDOVF, DEVICE=3390
*
SEGM  NAME=SEG6IX1, PARENT=0, BYTES=8
FIELD NAME=( IXEMP6X1, SEQ, M ), BYTES=8, START=1
LCHILD NAME=( LANGUAGE, AIHDAM ), INDEX=IXEMP6, PTR=SNGL
*
DBDGEN
FINISH
END

```


B Tracing Interface Processing

When you issue a report request, the Interface communicates with the IMS DBMS through DL/I calls that it generates on your behalf. This appendix discusses trace facilities that are available to help you solve problems:

- IMS provides a trace, DLITRACE, that traps DL/I calls, displays all their parameters, and shows the records returned by IMS and the corresponding status codes (see Section B.1, *DLITRACE*).
- The Interface provides three traces: FSTRACE, FSTRACE2, and FSTRACE4 (see Section B.2, *Interface Traces*).

FSTRACE shows the state of the PCB before and after each DL/I call; it also provides, for each call, the parameters, key feedback area, SSA buffer, return code, and I/O buffer contents.

FSTRACE2 shows the Userid, Filename and return code used and returned by the exit.

FSTRACE4 provides a dump of the SSA buffer for each SSA generated by a given request. You can use this trace to see whether your requests produce qualified SSAs. In the DBCTL environment, FSTRACE4 provides additional statistics.

B.1 DLITRACE

To trigger the IMS DLITRACE facility, you must allocate two ddnames in your JCL or CLIST.

1. Allocate ddname DFSVSAMP to a partitioned dataset or sequential file with record length 80 and load the VSAM buffer pool size information into this dataset or file. (You should first find out whether your site has a pre-defined set of buffer pools.) The following is a sample allocation

```
//DFSVSAMP DD DSN=highlvl.DFSVSAMP(BUFPOOL),DISP=SHR
```

where:

highlvl Is the high-level qualifier for your VSAM buffer pool dataset.

This allocation indicates that member BUFPOOL in the DFSVSAMP dataset contains the VSAM buffer pool size information. For example:

```
8192,21                    /* 21 BUFFERS OF SIZE 8192
4096,21                    /*
2048,21                    /*
1024,30                    /*
512,21                    /*
IOBF=(2048,4,Y,Y)         /*
IOBF=(6144,4,Y,Y)         /* IO BUFFERS OF THIS SIZE
IOBF=(8192,4,Y,Y)         /*
OPTION,VSAMFIX=(BFR,IOB)
```

2. To turn on the trace, add the following line to the end of the VSAM buffer pool information

```
DLITRACE DDNAME=ddname
```

where:

ddname Is any ddname. Allocate this ddname to SYSOUT or to a file with LRECL 80. The trace output is sent to the file or SYSOUT class you allocate. If you allocate the trace to a file, keep in mind that trace output can be voluminous.

Suppose you choose the ddname DDTRACE by adding this line to your VSAM buffer pool file:

```
DLITRACE DDNAME=DDTRACE
```

To send the trace output to SYSOUT, include the following allocation in your JCL:

```
//DDTRACE DD SYSOUT=*
```

Note:

- This trace facility is valid for DLI or batch mode (see Chapter 5, *Environments*). To trace BMP mode, the operator must issue a command from the MVS console to start a PSB trace.
- You must allocate the ddnames in the address space that actually issues the DL/I calls, either the XMI server address space or the FOCUS address space, depending on your environment (see Chapter 5, *Environments*).

B.2 Interface Traces

When you submit a report request, the Interface generates calls to the IMS DBMS on your behalf. You can use the Interface trace facilities to view these calls and their results.

A trace is helpful for debugging a procedure or for Interface performance analysis. The FSTRACE, FSTRACE2, and FSTRACE4 facilities are easy to invoke and require no changes to either the Interface or user request. The trace facilities have no effect on how the Interface functions.

The Interface provides the following trace facilities:

- FSTRACE captures or displays the PCB, the call parameters, and the SSA buffer prior to each DL/I call, and it shows the PCB, key feedback area, return code, and a dump of the I/O buffer contents after each DL/I call. Section B.2.4, *FSTRACE Example*, includes an example of FSTRACE output.
- FSTRACE2 shows the Userid, Filename and return code used and returned by the exit.
- FSTRACE4 captures or displays each SSA generated for a request. Chapter 4, *Reporting Efficiencies*, includes several examples of FSTRACE4 output.

In the DBCTL environment, FSTRACE4 also provides statistics about the number of calls, reads, enqueues, and buffers. Section B.2.5, *FSTRACE4 Statistics in the DBCTL Environment*, provides an example of the statistics generated in the DBCTL environment.

To activate one or both of the trace facilities, allocate the corresponding trace output as described in Section B.2.1, *Allocating Interface Traces*.

Note: FSTRACE, FSTRACE2 and FSTRACE4 are intended for use in query optimization and problem debugging. Do not write application programs that depend on the format or content of any trace, as these may change in later releases.

B.2.1 Allocating Interface Traces

You can use FSTRACE, FSTRACE2, and FSTRACE4 in conjunction with all report requests. You can store trace output in a sequential dataset or display it online at the terminal. Use the system editor to view the file containing the trace information.

To capture trace data in a file, issue the appropriate statement from the FOCUS command level

```
{MVS|TSO} ALLOC F(ddname) DA('userid.ddname') SHR REU LRECL(80) RECFM(F)
```

or

```
DYNAM ALLOC FILE ddname DATASET userid.ddname -  
SHR REUSE LRECL 80 RECFM F
```

or

```
CMS FILEDEF ddname DISK ddname DATA A  
(LRECL 80 RECFM F
```

where:

ddname Is FSTRACE, FSTRACE2, or FSTRACE4.

To display trace data online, specify the appropriate statement from the FOCUS command level:

```
{MVS|TSO} ALLOC F(ddname) DA(*) SHR REU
```

or

```
DYNAM ALLOC FILE ddname DA *
```

or

```
CMS FILEDEF ddname TERMINAL
```

where:

ddname Is FSTRACE, FSTRACE2, or FSTRACE4.

B.2.2 Disabling Interface Traces

To disable FSTRACE, FSTRACE2 or FSTRACE4, clear the associated allocation

```
{MVS|TSO} FREE F(ddname)
```

or

```
DYNAM FREE FILE ddname
```

or

```
CMS FILEDEF ddname CLEAR
```

where:

ddname Is FSTRACE, FSTRACE2 or FSTRACE4.

B.2.3 Batch Trace Allocation

In batch, you can write trace results to a sequential file, a partitioned dataset, or, as in the following sample allocation, to SYSOUT

```
//ddname DD SYSOUT=*,DCB=BLKSIZE=80
```

where:

ddname Is FSTRACE, FSTRACE2, or FSTRACE4.

The following example writes the trace results to an MVS sequential dataset

```
//FSBATCH EXEC PGM=FOCUS
//ddname DD DISP=(NEW,CATLG,KEEP),DSN=userid.ddname,
// UNIT=SYSDA,VOL=SER=volid,SPACE=(TRK,(p,s)),
// DCB=(LRECL=80,BLKSIZE=1600,RECFM=F)
```

where:

userid Is the user ID.

ddname Is FSTRACE, FSTRACE2, or FSTRACE4.

volid Is the volume ID of the dataset that will receive the trace output.

p Is the number of tracks in the primary space allocation.

s Is the number of tracks in the secondary space allocation.

B.2.4 FSTRACE Example

This TSO session traces the retrieval process for a report request. FSTRACE is allocated before the report request is issued; its results display online.

The request is as follows:

```
> > DYNAM ALLOC FILE FSTRACE DA *
> > SET IMS=NEW
> > TSO IMS SET IMSPZP 00
> > TSO IMS SET DBCTL ON
> > TSO IMS SET PSB FOCSD
> > TABLE FILE DI21PART
> PRINT PARTKEY STANKEY
> WHERE PARTKEY IN ('02AN960C10','02JAN1N976B')
> END
```

A trace can produce extensive output; therefore, you may want to use a RECORDLIMIT test in your request to reduce the amount of redundant information you will have to examine.

The following annotated example illustrates FSTRACE results for the request. Because this particular request includes the SET commands that turn DBCTL on and select the PSB, the calls to initialize the DRA (Database Resource Adapter) and schedule the PSB are included in the trace.

Trace output is separated into sections for each call to DBCTL; dotted lines mark the beginning and end of every section. Each individual section is also separated into two parts. The top part depicts information that the Interface passes to DBCTL; the bottom part presents the results of the call. The numbers on the left refer to explanatory notes listed at the end of the trace.

```

-----Calling DBCTL-----
1.  Function = Initialize DRA
    ImdSrv=5439070
    usid=USER1      ,sufx=00,susp=5486ad0,resm=5486af8,cntl=5486a28,tstx=0
-----
-----Back from DBCTL-----
    Function = Initialize DRA
2.  ctok=1e8e8,retc=0,dbct=IMS3,usid=USER1      ,mint=0,maxt=0,timo=999
-----
-----Control Exit Info-----
    ImdSrv=5439070
    func=2,sfnc=0,retc=0,rcod=0
    dbct=IMS3,jobn=IMS      ,rslt=0,rgty= 2,mtcb=1,dsid= 62
    idtk=z::,.:9:,rsen=      ,jsid=      ,mxn2=0,min2=0,hit2=0,tim2=0,mxt2=0
-----
-----Calling DBCTL-----
3.  Function = Schedule PSB
    ImdSrv=5439070, ImdUsr=5456848
    ctok=1e8e8,ttok=5456848,psb=FOCSD      ,wrth= 1,cflg= 0,alan= 2,stat=0
-----
-----Back from DBCTL-----
    Function = Schedule PSB
    retc=0,ctk2=5457aa8,pcbl=35cc0,fpcb=35ccc,iosz=600,plan= 6,mkey=33
-----
** Before IMS call
4.  There are 4 args
    cmd is (GU      )
    PCB: DI21PART00 GO
5.  ssa 1
    D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2      *PARTROOT*-(PARTK*
    C5E8407E 40F0F2C1 D5F9F6F0 C3F1F040      *EY = 02AN960C10 *
    40404040 40405D40      *      )      *
-----
-----Calling DBCTL-----
    Function = DLI request
    ImdSrv=5439070, ImdUsr=5456848
    ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f9e8,alan= 2
-----
-----Back from DBCTL-----
    Function = DLI request
    ImdSrv=5439070, ImdUsr=5456848
    retc=0,segl=50
-----
** After IMS call .
6.  PCB: DI21PART01 GO
    feedback len is (17)
    feedback area:
    F0F2C1D5 F9F6F0C3 F1F04040 40404040      *02AN960C10      *
    40      *      *

```

Tracing Interface Processing

```
7.   io buff:
    F0F2C1D5 F9F6F0C3 F1F04040 40404040      *02AN960C10      *
    40404040 40404040 4040E6C1 E2C8C5D9      *           WASHER*
    40404040 40404040 40404040 40404040      *           *
    40400000 00000000 00000000 00000000      *           *
    00000000 00000000 00000000 00000000      *           *
    ** Before IMS call

8.   There are 5 args
    cmd is (GU )
    PCB: DI21PART01 GO

9.   ssa 1: PARTROOT*U
    ssa 2
    E2E3C1D5 C9D5C6D6 5C604040      *STANINFO*-      *
    -----Calling DBCTL-----
    Function = DLI request
    ImdSrv=5439070, ImdUsr=5456848
    ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f710,alan= 2
    -----
    -----Back from DBCTL-----
    Function = DLI request
    ImdSrv=5439070, ImdUsr=5456848
    retc=0,segl=85
    -----
    ** After IMS call **
    PCB: DI21PART02 GO
    feedback len is (19)
    feedback area:
    F0F2C1D5 F9F6F0C3 F1F04040 40404040      *02AN960C10      *
    40F0F2      * 02      *
    io buff:
    F0F24040 40404040 40404040 40404040      *02      *
    4040F7F4 F2404040 40404040 40404040      * 742      *
    40404040 40404040 40404040 404040F1      *           1*
    F2F0F040 40F1F440 40404040 40F0F6C3      *200 14      06C*
    40404040 4040F0F6 C3404040 40404040      *           06C      *
    ** Before IMS call

10.  There are 5 args
    cmd is (GN )
    PCB: DI21PART02 GO
    ssa 1: PARTROOT*U
    ssa 2
    E2E3C1D5 C9D5C6D6 5C604040      *STANINFO*-      *
    -----Calling DBCTL-----
    Function = DLI request
    ImdSrv=5439070, ImdUsr=5456848
    ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f710,alan= 2
    -----
    -----Back from DBCTL-----
    Function = DLI request
    ImdSrv=5439070, ImdUsr=5456848
    retc=0,segl=0
    -----
```

```

** After IMS call
11. PCB: DI21PART01GEGO
feedback len is (17)
feedback area:
F0F2C1D5 F9F6F0C3 F1F04040 40404040      *02AN960C10      *
40                                           *                *
  io buff:
F0F24040 40404040 40404040 40404040      *02                *
4040F7F4 F2404040 40404040 40404040      * 742              *
40404040 40404040 40404040 404040F1      *                  1*
F2F0F040 40F1F440 40404040 40F0F6C3      *200 14           06C*
40404040 4040F0F6 C3404040 40404040      *      06C        *
** Before IMS call **
There are 4 args
cmd is (GN )
  PCB: DI21PART01GEGO
  ssa 1
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2      *PARTROOT*-(PARTK*
C5E8407E 40F0F2C1 D5F9F6F0 C3F1F040      *EY = 02AN960C10 *
40404040 40405D40                          *      )          *
-----Calling DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f9e8,alan= 2
-----
-----Back from DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
retc=0,segl=0
-----
** After IMS call **
  PCB: DI21PART00GEGO
feedback len is (0)
feedback area:
  io buff:
F0F24040 40404040 40404040 40404040      *02                *
4040F7F4 F2404040 40404040 40404040      * 742              *
40404040 40404040 40404040 404040F1      *                  1*
F2F0F040 40F1F440 40404040 40F0F6C3      *200 14           06C*
40404040 4040F0F6 C3404040 40404040      *      06C        *
** Before IMS call
12. There are 4 args
cmd is (GU )
  PCB: DI21PART00GEGO
  ssa 1
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2      *PARTROOT*-(PARTK*
C5E8407E 40F0F2D1 C1D5F1D5 F9F7F6C2      *EY = 02JAN1N976B*
40404040 40405D40                          *      )          *
-----Calling DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f9e8,alan= 2
-----

```

Tracing Interface Processing

```
-----Back from DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
retc=0,segl=50
-----
** After IMS call **
  PCB: DI21PART01 GO
feedback len is (17)
feedback area:
F0F2D1C1 D5F1D5F9 F7F6C240 40404040      *02JAN1N976B      *
40                                           *                *
  io buff:
F0F2D1C1 D5F1D5F9 F7F6C240 40404040      *02JAN1N976B      *
40404040 40404040 4040C4C9 D6C4C540      *                DIODE *
C3D6C4C5 60C14040 40404040 40404040      *CODE-A          *
4040F040 40F1F440 40404040 40F0F6C3      * 0 14          06C*
40404040 4040F0F6 C3404040 40404040      *                06C  *
** Before IMS call **
There are 5 args
cmd is (GU )
  PCB: DI21PART01 GO
  ssa 1: PARTROOT*U
  ssa 2
E2E3C1D5 C9D5C6D6 5C604040      *STANINFO*-      *
-----Calling DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f710,alan= 2
-----
-----Back from DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
retc=0,segl=85
-----
** After IMS call **
  PCB: DI21PART02 GO
feedback len is (19)
feedback area:
F0F2D1C1 D5F1D5F9 F7F6C240 40404040      *02JAN1N976B      *
40F0F2                                           * 02                *
  io buff:
F0F24040 40404040 40404040 40404040      *02                *
4040F7F4 F2404040 40404040 40404040      * 742              *
40404040 40404040 40404040 404040F1      *                1 *
F2F0F040 40F7F240 40404040 40F0F6C3      *200 72          06C*
40404040 4040F0F6 C3404040 40404040      *                06C  *
** Before IMS call **
There are 5 args
cmd is (GN )
  PCB: DI21PART02 GO
  ssa 1: PARTROOT*U
  ssa 2
E2E3C1D5 C9D5C6D6 5C604040      *STANINFO*-      *
```

```

-----Calling DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f710,alan= 2
-----
-----Back from DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
retc=0,segl=0
-----
** After IMS call **
PCB: DI21PART01GEGO
feedback len is (17)
feedback area:
F0F2D1C1 D5F1D5F9 F7F6C240 40404040      *02JAN1N976B      *
40                                     *                *
io buff:
F0F24040 40404040 40404040 40404040      *02                *
4040F7F4 F2404040 40404040 40404040      * 742              *
40404040 40404040 40404040 404040F1      *                  1*
F2F0F040 40F7F240 40404040 40F0F6C3     *200 72           06C*
40404040 4040F0F6 C3404040 40404040     *          06C      *
** Before IMS call **
There are 4 args
cmd is (GN )
PCB: DI21PART01GEGO
ssa 1
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2      *PARTROOT*-(PARTK*
C5E8407E 40F0F2D1 C1D5F1D5 F9F7F6C2     *EY = 02JAN1N976B*
40404040 40405D40                          *          )          *
-----Calling DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
ctok=1e8e8,ctk2=5457aa8,ttok=5456848,clst=1f9e8,alan= 2
-----
-----Back from DBCTL-----
Function = DLI request
ImdSrv=5439070, ImdUsr=5456848
retc=0,segl=0
-----
** After IMS call **
PCB: DI21PART00GEGO
feedback len is (0)
feedback area:
io buff:
F0F24040 40404040 40404040 40404040      *02                *
4040F7F4 F2404040 40404040 40404040      * 742              *
40404040 40404040 40404040 404040F1      *                  1*
F2F0F040 40F7F240 40404040 40F0F6C3     *200 72           06C*
40404040 4040F0F6 C3404040 40404040     *          06C      *

```

Tracing Interface Processing

```
-----Calling DBCTL-----
13. Function = Prepare for Commit
   ImdSrv=7e080c8, ImdUsr=7e18578
   ctok=21840,ctk2=7ebb530,ttok=7e18578,stcd=7d52fb0
-----
-----Back from DBCTL-----
Function = Prepare for Commit
retc=0,stcd=3,
-----
-----Calling DBCTL-----
Function = Commit - Terminate
   ImdSrv=7e080c8, ImdUsr=7e18578
   ctok=21840,ctk2=7ebb530,ttok=7e18578,rtok=c4d6c3e2
-----
-----Back from DBCTL-----
Function = Commit - Terminate
retc=0
-----
-----Calling DBCTL-----
Function = Terminate thread
   ImdSrv=7e080c8, ImdUsr=7e18578
   ctok=21840,ctk2=7ebb530,ttok=7e18578-----
-----Back from DBCTL-----
Function = Terminate thread
retc=0
-----
NUMBER OF RECORDS IN TABLE=          2  LINES=          2

PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

The trace shows the following:

1. The call to initialize the DRA; the Interface passes both the userid (USER1) and the suffix for the DRA Startup Table (00) to DBCTL.
2. DBCTL sends back a return code (retc) of 0 signifying that the DRA was initialized with no problem. A non-zero return code at this stage indicates a need to execute the IMDTST JCL described in Appendix D, *Installation Instructions*.
3. The call to schedule the PSB; the Interface passes the name of the PSB, FOCSD, to DBCTL. The return code of 0 after the call indicates that the PSB was scheduled with no problem.
4. The number of arguments for the first DL/I call to IMS. Each DL/I call requires the following arguments: the function to be performed (GU or GN), the PCB, an I/O area, and, if relevant, one or more SSAs. The number of SSAs depends on the nature of the request and the number of segments involved.

This call is a GU call. The PCB contains the database name, DI21PART, and the PROCOPT, GO.

5. The SSA for this call. There is one SSA to locate the root segment with PARTKEY=02AN960C10. The trace provides the SSA in the form of a dump of the SSA buffer.

A dump has a left side and a right side. The left side shows the hexadecimal values in the memory locations. The right side, delimited by *'s, contains the corresponding alphanumeric characters. Since not all hexadecimal values represent printable alphanumeric characters, those parts of the dump that correspond to non-printable characters display as blanks on the right side. In particular, if a comparison value in an SSA is a packed number, it displays as blank on the right-hand side of the dump.

In this case, PARTKEY is an alphanumeric field, so the comparison value, 02AN960C10, is also alphanumeric and displays in the dump. The Interface constructs a qualified SSA that includes the segment name, PARTROOT, and the logical expression, PARTKEY=02AN960C10.

When there are no selection criteria for a segment, the Interface constructs an SSA that contains only the segment name.

6. The PCB after the IMS call. The segment level (01) of the retrieved segment is now in the PCB, and the status code (2 blanks) indicates a successful retrieval. The key feedback area is also part of the PCB and contains the concatenated key of each segment in the hierarchical path to the retrieved segment. (Since only the root segment was retrieved in this call, the key feedback area contains the retrieved PARTKEY value.)
7. A dump of the I/O buffer after the DL/I call. This buffer contains the retrieved segment. For instructions about reading the dump, see item 5.
8. The start of the next call to IMS, a GU call to retrieve the first child of the PARTROOT segment retrieved in the previous call. This call has five arguments because there are now two SSAs: one for each segment.
9. The SSAs for this call. The first SSA uses the U command code to stay within the chain of children of the previously-retrieved PARTROOT segment.

The second SSA is for the STANINFO segment (the FOCUS request prints the STANKEY field from this segment). Since the request contains no selection criteria on this segment, the SSA is unqualified and contains only the segment name. This call retrieves a STANINFO segment with STANKEY = 02 (see the feedback area and I/O buffer).

10. The third DL/I call, a GN call to retrieve the next child of the previously-retrieved PARTROOT segment. The SSAs are the same as those in item 9.
11. The PCB after the third call to IMS. The GE status code in the PCB indicates that no segment was found that satisfied the search criteria.
12. A GU call to locate a PARTROOT segment with PARTKEY=02JAN1N976B. The entire sequence of calls described in the previous items repeats for this value of PARTKEY.

13. The cleanup at the end of the request. The cleanup process terminates the DBCTL threads and unchedules the PSB.

The following report is generated:

PAGE	1
PARTKEY	STANKEY
-----	-----
02AN960C10	02
02JAN1N976B	02

B.2.5 FSTRACE4 Statistics in the DBCTL Environment

When you allocate FSTRACE4 in the DBCTL environment, the trace provides statistics about the number of calls made, enqueues, waits, and buffers. The following example illustrates FSTRACE4 results in this environment:

```
> > DYNAM ALLOC FILE FSTRACE4 DA *
> > TABLE FILE DI21PART
> PRINT PARTKEY STANKEY
> WHERE PARTKEY IN ('01AN960C10','02JAN1N976B')
> END
```

The resulting trace provides the SSA buffer and the UOR statistics:

```
set up SSA-Q:
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2 *PARTROOT*-(PARTK*
C5E8407E 40F0F1C1 D5F9F6F0 C3F1F040 *EY = 01AN960C10 *
40404040 40405D40 * ) *
set up SSA-Q:
D7C1D9E3 D9D6D6E3 5C604DD7 C1D9E3D2 *PARTROOT*-(PARTK*
C5E8407E 40F0F2D1 C1D5F1D5 F9F7F6C2 *EY = 02JAN1N976B*
40404040 40405D40 * ) *
set up SSA-Q:
E2E3C1D5 C9D5C6D6 5C604040 *STANINFO*- *
-----UOR Statistics-----
GU calls=3,GN calls=2,GNP calls=0,GHU calls=0,GHN calls=0,GHNP calls=0,
ISRT calls=0,DELT calls=0,REPL calls=0,Total DLI calls=5,
Test ENQ=0,Waits for Test ENQ=0,Test DEQ=0,Update ENQ=0,Waits for Updt
ENQ=0,U
P
Exclusive ENQ=0,Waits on Exclusive ENQ=0,Exclusive DEQ=0,
DEDB Calls=0,DEDB Reads=0,Overflow Buffers Used=0,UOW Contentions=0,
Waits for DEDB Buffers=0,Schedule Seq Number=998,
Elapsed UOR CPUTIME (Timer units)=209.
-----
```

If your request seems to take a long time, you may want to allocate FSTRACE4 to see if any of the numbers are very high. Enqueues become a factor depending on the PROCOPT you use.

C Release Dependent Interface Features

In FOCUS Release 7.0, certain default values have changed. This appendix explains features and settings that have changed from previous releases. It also describes Interface environmental commands that enable you to set parameters that govern your FOCUS session. You can view the settings in effect at any time by issuing the IMS SET ? query command. You can issue SET commands either from the command line, in the MSO or FOCUS profile, or in a FOCEXEC.

This appendix includes the following topics:

- The SET IMS command (see Section C.1, *The SET IMS Command*).
- Interface environmental commands (see Section C.2, *Interface Environmental Commands*).
- Secondary index definitions that existed prior to the Auto Index Selection feature (see Section C.3, *Describing a Secondary Index Without Auto Index Selection*).
- Fixed format FOCPSBs (see Section C.4, *Fixed Format FOCPSBs*).
- Access to the BMP extension (see Section C.5, *Accessing the BMP Extension*).
- IMS access from CMS (see Section C.6, *Accessing IMS Databases From CMS*).

C.1 The SET IMS Command

Because of the changes introduced in FOCUS Releases 6.8 and 7.0, two versions of the IMS/DB Interface are available. With the new version of the Interface, you have access to features such as comma-delimited FOCPSBs, the XMI server, and the Interface environmental commands described in Section C.2, *Interface Environmental Commands*.

The syntax is

```
SET IMS = {NEW|OLD}
```

where:

- | | |
|-----|--|
| NEW | Activates new features introduced in FOCUS Release 6.8 and 7.0. NEW is the default setting in FOCUS Release 7.0. |
| OLD | Provides access to only those Interface features that were available in FOCUS releases prior to 6.8. OLD was the default setting in FOCUS Release 6.8. |

Release Dependent Interface Features

When SET IMS=NEW, you have access to Interface new features such as comma-delimited FOCPSBs, the DBCTL environment (available with IMS 3.1 and up), the XMI server environment, Auto Index Selection, and the Interface environmental commands described in this appendix. When SET IMS=OLD, you invoke the Interface that was available with prior FOCUS releases, and you cannot take advantage of the new features.

If you are not sure which FOCUS release you are running, issue the following command:

```
? RELEASE
```

For an explanation of how to determine which IMS setting is actually in effect at any point during your session, see Section C.2.2, *The IMS SET ? Query Command*.

C.2 Interface Environmental Commands

When SET IMS=NEW, you have access to Interface environmental commands with which you can change certain parameters that control your FOCUS session.

To distinguish them from FOCUS commands such as the SET IMS command, Interface environmental commands include an MVS or TSO environmental qualifier and the keyword IMS.

The syntax is

```
{MVS|TSO} IMS SET env_command value
```

where:

env_command Is an environmental command.

value Is an acceptable value for the environmental command.

Although the commands are prefixed with the environmental qualifier TSO or MVS, the Interface, and not the operating system, handles them.

Note: The MVS and TSO environmental qualifiers are synonymous and can be used interchangeably in any MVS environment (for example, TSO, batch, or MSO).

C.2.1 Commands for Implementing the DBCTL Environment

From a TSO session or batch job, you invoke the DBCTL environment (available with IMS 3.1 and up, see Chapter 5, *Environments*) by issuing the following two commands in the order shown

```
{MVS|TSO} IMS SET IMSPZP {xx|00}
{MVS|TSO} IMS SET DBCTL {ON|OFF}
```

where:

xx Sets the two-character suffix for the DRA Startup Table, chosen during installation (see Appendix D, *Installation Instructions*). If you do not issue this command, the suffix defaults to 00.

ON Turns on the DBCTL environment.

OFF Turns off the DBCTL environment. OFF is the default in a TSO session.

Note: If you are running MSO:

- These settings are included in the MSO configuration file; see Appendix D, *Installation Instructions*.
- DBCTL ON is the default as long as DBCTL is available, indicated by the attribute DBCTL=START in the configuration file. You can then control the DBCTL setting using console commands, as described in Appendix D, *Installation Instructions*.

If you did not create an Access File to automatically select the PSB for your request (see Chapter 3, *Creating FOCUS Descriptions*) or to change the selected PSB, issue the following command

```
{MVS|TSO} IMS SET PSB psbname
```

where:

psbname Is the name of the FOCPSB library member to use. This name must be identical to the name of the actual PSB that IMS will access. If the member does not exist, the following error message is generated:

```
(FOC4261) FOCPSB MEMBER NOT FOUND: psbname
```

Note:

- All participating files in a join must use the same PSB. Any attempt to join files that require different PSBs generates the following error message:


```
(FOC4295) ACCESS POINTS TO DIFFERENT PSBS IN JOIN
```
- The IMS SET PSB command supersedes the PSB attribute in the Access File.

C.2.2 The IMS SET ? Query Command

If SET IMS=NEW, the default, you can issue the IMS SET ? command at any time during your session to display Interface defaults and current settings.

The syntax is:

```
{MVS|TSO} IMS SET ?
```

For example:

```
> > TSO IMS SET DBCTL ON
> > TSO IMS SET IMSPZP 05
> > TSO IMS SET PSB TSTPSB01
> > TSO IMS SET ?
(FOC4280) IMSMODE      : DBCTL
(FOC4281) PSB          : TSTPSB01
(FOC4282) IMSSEC       : OFF
(FOC4283) IMSCLASS     :
(FOC4284) IMSPZP       : 05
```

Note:

- IMSSEC and IMSCLASS are security settings available through MSO (see Chapter 6, *Security*).
- IMSMODE values are:

Value	Mode
DBCTL	DBCTL
REMOTE	BMP
LOCAL	DLI

For more information on modes, see Chapter 5, *Environments*.

- If you issue the IMS SET ? query command with SET IMS=OLD in effect, no settings are displayed, and you are returned to the FOCUS prompt. Therefore, you can determine which IMS setting (OLD or NEW) is in effect by whether the IMS SET ? query command produces a display of current settings.

C.3 Describing a Secondary Index Without Auto Index Selection

Starting in FOCUS Release 7.0.6, you can create one Master File and FOCPSB that describe the primary index and multiple secondary indexes for a database. The Interface Auto Index Selection feature can then determine the appropriate index to use for each request. For information on this feature, see Chapter 2, *IMS Overview and Mapping Concepts*, and Chapter 3, *Creating FOCUS Descriptions*.

In prior releases, each Master File was limited to describing either the primary index or one secondary index. The application programmer then had to decide which index to use for each request and implement that decision by referencing the appropriate Master File in each request. This technique is still supported, and this section describes it.

Using IMS secondary indexes, you can retrieve records in order of a field other than the key field. (A secondary index is itself a database with its own DBD.) The DBD for a database that uses a secondary index includes an XDFLD statement that assigns a field name to the index.

If a PCB includes the parameter PROCSEQ=indexname, the named index is used as the main entry point into the database.

The Interface can take advantage of one secondary index for each Master File if either of the following conditions exists:

- The index is specified by the PROCSEQ parameter in the PCB.
- The index targets the root segment in the Master File.

To use the index, in the Master File you must do the following:

- Describe the secondary index as the key field for the root segment. To do so, specify ALIAS=XDFLDname.KEY, where XDFLDname is the name assigned by the XDFLD record in the DBD

```
ALIAS=XDFLDname . KEY
```

where:

`XDFLDname` Indicates the name assigned by the XDFLD record in the DBD.

- Describe this field as the *last* field of the root segment.
- Describe the actual key of the root segment with the suffix IMS (a FOCUS segment can have at most one field defined with the suffix KEY)

```
ALIAS=KEYname . IMS
```

where:

`KEYname` Indicates the IMS name of the key field as described in the DBD.

When the Interface generates SSAs for retrieval, it will use the XDFLD as the key. It can still use the real key field in DL/I calls since the real key is defined as a search field (suffix IMS).

Release Dependent Interface Features

The following diagram illustrates how to describe a secondary index in the Master File:

<p>1) IMS PSB</p> <p>PCB DBDNAME=DBD1 PROCSEQ=SINDEX</p>	<p>2) IMS DBD</p> <p>For the Secondary Index</p> <p>DBD NAME=SINDEX ACCESS=INDEX</p> <p>SEGM NAME=SEGX FIELD NAME=(UNAME,SEQ,U) LCHILD NAME=(SEG1,DBD1) INDEX=XNAME</p>	<p>3) Master File Description</p> <p>FILENAME=DBD1</p>
<p>SENSEG NAME=SEG1 PARENT=0</p> <p>SENSEG NAME=SEG2 PARENT=SEG1</p>	<p>For the database;</p> <p>DBD NAME=DBD1</p> <p>SEGM NAME=SEG1 PARENT=0 FIELD NAME=(SSN,SEQ,U)</p> <p>XDFLD NAME=XNAME</p> <p>SEGM NAME=SEG2 PARENT=SEG1</p>	<p>SEGNAME=SEG1,SEGTYPE=S2</p> <p>FIELDNAME=SSNO ALIAS=SSN.IMS</p> <p>FIELDNAME=IX ALIAS=XNAME.KEY</p> <p>SEGNAME=SEG2 PARENT=SEG1</p>

1. Is the IMS PCB. It is sensitive to SEG1 and SEG2. PROCSEQ=SINDEX means that a secondary index is used as the main entry point into the database, and that the DBD describing this index is named SINDEX.
2. Shows the DBDs for the database and the index:

The index DBD specifies that the index is for SEG1 in DBD1 (the root, as required) and that its name is XNAME.

The database DBD contains the XDFLD statement in the root segment. It specifies the name XNAME. The key for the root segment is SSN.
3. Is the Master File. The index is the last field specified in the root segment, SEG1, and it is described as the key to the segment (ALIAS=XNAME.KEY). XNAME is the name from the XDFLD statement in the DBD.

The field that is the actual key to the root segment, SSN from the DBD, is described as a search field (ALIAS=SSN.IMS).

Describing a Secondary Index Without Auto Index Selection

If necessary, you can add an additional PCB to the PSB for use with a secondary index. Include the parameter PROCSEQ=indexDBDname in this PCB, and create a corresponding Master File according to the previous instructions. You can create a separate PCB and Master File for each secondary index you need.

The following example shows a secondary index defined in the PATDB01 DBD. The DBD for the secondary index, PATDBIX1, is as follows:

```
PRINT NOGEN
DBD      NAME=PATDBIX1,ACCESS=INDEX
DATASET DD1=PATDBIX1,DEVICE=3380
*
SEGM NAME=SEGIX1,PARENT=0,BYTES=28
      FIELD NAME=(IXNAMEIX,SEQ,U),BYTES=28,START=1
      LCHILD NAME=(PATINFO,PATDB01),INDEX=IXNAME,PTR=SNGL
DBDGEN
FINISH
END
```

In the TSTPSB01 PSB, the PCB that accesses this index includes the parameter PROCSEQ=PATDBIX1:

```
PCB      TYPE=TP,MODIFY=YES,EXPRESS=YES
PCB      TYPE=TP,EXPRESS=NO,MODIFY=YES,SAMETRM=YES
PCB      TYPE=DB,DBDNAME=PATDB01,PROCOPT=GO,KEYLEN=9
SENSEG   NAME=PATINFO,PARENT=0
PCB      TYPE=DB,DBDNAME=PATDB01,PROCOPT=GO,KEYLEN=28,PROCSEQ=PATDBIX1
SENSEG   NAME=PATINFO,PARENT=0
PSBGEN   LANG=COBOL,PSBNAME=TSTPSB01,CMPAT=YES
END
```

In the FOCPSB, the corresponding PCB record assigns the name PATINIX to the Master File:

```
FOCPSB=EXTENDED,$
PCBNAME=          ,PCBTYPE=TERM,$
PCBNAME=          ,PCBTYPE=TERM,$
PCBNAME=          ,PCBTYPE=TERM,$
PCBNAME=PATINFO  ,PCBTYPE=DB,$
PCBNAME=PATINIX  ,PCBTYPE=DB,$
```

Release Dependent Interface Features

The DBD for the PATDB01 database includes an XDFLD record that assigns the name IXNAME to index PATDBIX1. The index searches the database in order of last name and first name and is targeted to the root segment:

```
PRINT NOGEN
DBD      NAME=PATDB01,ACCESS=(HIDAM,VSAM)
DATASET DD1=PATDB01,DEVICE=3380,BLOCK=4096,SCAN=5
*
SEGM NAME=PATINFO,PTR=H,PARENT=0,BYTES=233
  FIELD NAME=(SSN,SEQ,U),BYTES=9,START=26,TYPE=C
  LCHILD NAME=(SEGIX,PATDBIX),PTR=INDX
  FIELD NAME=SEQFIELD,BYTES=6,START=1,TYPE=C
  FIELD NAME=REVSEQ,BYTES=6,START=7,TYPE=C
  FIELD NAME=SSNALPHA,BYTES=9,START=35,TYPE=C
  FIELD NAME=EMPID,BYTES=12,START=44,TYPE=C
  FIELD NAME=LNAME,BYTES=12,START=56,TYPE=C
  FIELD NAME=FNAME,BYTES=12,START=68,TYPE=C
  FIELD NAME=ADMDATE,BYTES=8,START=89,TYPE=C
  FIELD NAME=PATID,BYTES=10,START=176,TYPE=C
  FIELD NAME=/SX1
*
  LCHILD NAME=(SEGIX1,PATDBIX1),PTR=INDX
  XDFLD  NAME=IXNAME,SRCH=(LNAME,FNAME),                                X
        SUBSEQ=/SX1,NULLVAL=BLANK
*
  LCHILD NAME=(SEGIX2,PATDBIX2),PTR=INDX
  XDFLD  NAME=IXCOMP,SRCH=(SSNALPHA,EMPID,LNAME)
*
  LCHILD NAME=(SEGIX3,PATDBIX3),PTR=INDX
  XDFLD  NAME=IXADMD,SRCH=(ADMDATE),                                    X
        SUBSEQ=/SX1,NULLVAL=BLANK
*
DBDGEN
FINISH
END
```

Describing a Secondary Index Without Auto Index Selection

In the PATINIX Master File, the last field described is the index field, and it has ALIAS=IXNAME.KEY. Notice that the real key to the segment, SSN, is defined as a search field (ALIAS=SSN.IMS):

```

FILE=PATDB01 , SUFFIX=IMS , $
SEGNAME=PATINFO , PARENT= , SEGTYPE=S1 , $
FIELD=SEQFIELD      , ALIAS=SEQFIELD.IMS      , USAGE=A06      , ACTUAL=A06 , $
FIELD=REVSEQ        , ALIAS=REVSEQ.IMS      , USAGE=A06      , ACTUAL=A06 , $
FIELD=SALARY        , ALIAS=                , USAGE=A08      , ACTUAL=A08 , $
FIELD=OT_HR_PAY     , ALIAS=                , USAGE=A05      , ACTUAL=A05 , $

FIELD=SSN           , ALIAS=SSN.IMS        , USAGE=A09      , ACTUAL=A09 , $

FIELD=SSNALPHA     , ALIAS=SSNALPHA.IMS  , USAGE=A09      , ACTUAL=A09 , $
FIELD=EMPLOYEEID   , ALIAS=EMPID.IMS     , USAGE=A12      , ACTUAL=A12 , $
FIELD=LAST_NAME    , ALIAS=LNAME.IMS     , USAGE=A12      , ACTUAL=A12 , $
FIELD=FIRST_NAME   , ALIAS=FNAME.IMS     , USAGE=A12      , ACTUAL=A12 , $
FIELD=DATE_OF_BIRTH , ALIAS=              , USAGE=A08      , ACTUAL=A08 , $
FIELD=RACE          , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $
FIELD=ADMIT_DATE   , ALIAS=ADMDATE.IMS  , USAGE=A08      , ACTUAL=A08 , $
FIELD=ADMIT_TYPE   , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $
FIELD=DISPOSITION  , ALIAS=              , USAGE=A02      , ACTUAL=A02 , $
FIELD=TRANSFER_DATE , ALIAS=              , USAGE=A08      , ACTUAL=A08 , $
FIELD=ALLERGY1     , ALIAS=              , USAGE=A15      , ACTUAL=A15 , $
FIELD=ALLERGY2     , ALIAS=              , USAGE=A15      , ACTUAL=A15 , $
FIELD=ALLERGY3     , ALIAS=              , USAGE=A15      , ACTUAL=A15 , $
FIELD=ALLERGY4     , ALIAS=              , USAGE=A15      , ACTUAL=A15 , $
FIELD=HOUSING      , ALIAS=              , USAGE=A03      , ACTUAL=A03 , $
FIELD=RPR          , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $
FIELD=URIN         , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $
FIELD=PRACTITIONAR , ALIAS=              , USAGE=A02      , ACTUAL=A02 , $
FIELD=SHIFT        , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $
FIELD=PATINET_ID   , ALIAS=PATID.IMS    , USAGE=A12      , ACTUAL=A12 , $
FIELD=WHO_ADDED    , ALIAS=              , USAGE=A10      , ACTUAL=A10 , $
FIELD=DATE_ADDED   , ALIAS=              , USAGE=A08      , ACTUAL=A08 , $
FIELD=WHO_EDITED   , ALIAS=              , USAGE=A10      , ACTUAL=A10 , $
FIELD=DATE_EDITED  , ALIAS=              , USAGE=A08      , ACTUAL=A08 , $
FIELD=STATION_ID   , ALIAS=              , USAGE=A12      , ACTUAL=A12 , $
FIELD=DIABETIC     , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $
FIELD=DIALYSIS     , ALIAS=              , USAGE=A01      , ACTUAL=A01 , $

FIELD=IXNAME       , ALIAS=IXNAME.KEY    , USAGE=A28      , ACTUAL=A28 , $

```

C.4 Fixed Format FOCPSBs

If SET IMS = OLD, you must use fixed format FOCPSBs. The FOCPSB file is an 80-character card image sequential file or PDS member (LRECL=80, RECFM=F or FB). Allocate it as ddname FOCPSB prior to accessing the database.

To associate a Master File with each PCB in the PSB, assign each PCB a filename. The records in the FOCPSB file are in the same order as the PCBs in the PSB. They have the following format:

Position	Contents
1	* = Comment line.
1-8	Master File name corresponding to a database PCB.
9	Ignored.
10-13	TERM Terminal PCB. DB Database PCB. SKIP PCB is to be ignored. blank Database PCB.
14-80	Comments or indication that PSB was for PL/I application program. If the PSB was originally generated for a PL/I application program, place the characters PL1, PLI, PL/1, or PL/I starting in or after position 14 of the first PCB, prior to any comments.

C.5 Accessing the BMP Extension

In the current FOCUS Release, the XMI server performs the function previously performed by the BMP extension (program FOCBMP). In order to access the BMP extension, you must SET IMS=OLD.

This is the required configuration for accessing IMS databases from CMS.

From the perspective of the end user, the only differences between using the XMI server and the BMP extension are the IMS setting and one parameter passed to the region controller program. Therefore, the explanations and diagrams that describe the XMI server in Chapter 5, *Environments*, all pertain to the BMP extension. Consult that chapter for a complete explanation. This section provides an example that accesses the BMP extension.

The BMP extension is an application (program FOCBMP) that intercepts DL/I calls from the IMS/DB Interface address space and issues them to IMS. Before you can invoke the Interface with this configuration, there must be a BMP extension job executing with the appropriate parameter settings and with an available PCB for the databases you want to access.

Therefore, access to the BMP extension is a two-part process. The two parts consist of:

1. Initiating a BMP extension.

To initiate a BMP extension, you run a region controller program and pass it parameters that describe the environment you need. The following chart lists the most common parameters:

Parameter	Value	Definition
Mode	BMP	Indicates Batch Message Processing mode. This mode accesses online IMS databases through the IMS/DC address space.
	DLI	Indicates local mode. This mode accesses IMS databases that you allocate locally.
Program	FOCBMP	Loads and accesses the BMP extension in the region controller address space.
PSB	psbname	Identifies the PSB to use.

One BMP extension can service multiple users, but each user must have a separate PCB. Therefore, to allow multiple users, the PSB for the BMP extension should contain duplicate PCBs.

You may not have to actually execute this step because if a BMP extension is already running with the correct parameters and an available PCB, you can use it.

2. Invoking the Interface once a BMP extension job is running.

When you use the BMP extension to access IMS databases, you must allocate a common communication file (to ddname FOCBMP) in both the FOCUS address space and the BMP extension address space. See Chapter 5, *Environments*, for a discussion of the communication file.

C.5.1 Example: Initiating a BMP Extension in DLI Mode

To initiate a BMP extension in DLI mode, submit the following JCL after editing it to conform to your site's standards and adding a JOB card. Note that you must allocate your IMS databases, DBDs, and PSBs in this JCL.

You do not have to submit this JCL if a BMP extension job that has an available PCB is already running with the following parameter settings

```
//DLIXMI EXEC PGM=DFSRR00,REGION=nn,PARM='DLI,FOCBMP,psbname'
//STEPLIB DD DSN=IMS.RESLIB,DISP=SHR
// DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
// DD DSN=prefix.IMS.LOAD,DISP=SHR
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//IMS DD DSN=IMS.PSBLIB,DISP=SHR
// DD DSN=IMS.DBDLIB,DISP=SHR
//DFSVSAMP DD DSN=IMS.DFSVSAMP(bufpool),DISP=SHR
//PATDB01 DD DSN=IMS.PATIENT.DB,DISP=SHR
//PATDBIX DD DSN=IMS.PATIENT.IX,DISP=SHR
//PATDBIX1 DD DSN=IMS.PATIENT.IX1,DISP=SHR
//PATDBIX2 DD DSN=IMS.PATIENT.IX2,DISP=SHR
//PATDBIX3 DD DSN=IMS.PATIENT.IX3,DISP=SHR
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR
//FOCPST DD DSN=prefix.IMS.FOCPSB(psbname),DISP=SHR
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//
```

where:

<i>nn</i>	Is the region size.
<i>psbname</i>	Is the name of the PSB to use.
<i>prefix</i>	Is the high-level qualifier for your site's FOCUS production libraries.
<i>bufpool</i>	Is the member that contains your VSAM buffer pool information.

Note:

- The ddnames allocated to the IMS databases are from the DD1 parameters in the relevant IMS DBDs. In the sample, the databases are the PATDB databases and indexes described in Appendix A, *Sample File Descriptions*.
- IMS.RESLIB is the IMS load library, IMS.DBDLIB is the IMS DBD library, and IMS.PSBLIB is the IMS PSB library.
- If you specify the parameter DBB instead of DLI on the EXEC card, you must include the ACB library in the allocation for ddname IMS. In addition, you must allocate a second dataset, IMS.ACBLIB, to ddname IMSACB.

- Ddname FOCBMP is allocated to the common communication file. You must allocate the same file when you invoke the Interface, as described in Chapter 5, *Environments*.
- Most error conditions detected by the BMP extension are reflected back to the originating user, where they are issued to the terminal. The SYSPRINT file allocated in the BMP extension job will contain only global error messages pertaining to the BMP extension environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the BMP extension are discussed in Chapter 5, *Environments*.

Once the BMP extension job is executing, you can invoke the Interface from a CLIST or from batch JCL, as described in Section C.5.2, *Invoking the Interface in the BMP Extension Environment*.

C.5.2 Invoking the Interface in the BMP Extension Environment

To invoke the Interface as a batch job, submit the following sample JCL after editing it to conform to your site's standards and adding a JOB card

```
//FOCXMI   EXEC PGM=FOCUS,REGION=nn
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//         DD DSN=prefix.IMS.LOAD,DISP=SHR
//         DD DSN=prefix.FUSELIB.LOAD,DISP=SHR
//ERRORS  DD DSN=prefix.ERRORS.DATA,DISP=SHR
//FOCBMP  DD DSN=prefix.FOCBMP.DATA,DISP=SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//MASTER DD DSN=prefix.MASTER.DATA,DISP=SHR
//SYSOUT  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
SET IMS=OLD
TABLE FILE ...
.
.
.
END
FIN
/*
//
```

where:

nn Is the region size.

prefix Is the high-level qualifier for your site's FOCUS production libraries.

Note:

- You must SET IMS=OLD in your FOCUS session, as shown in the sample.
- You can also invoke the Interface from a CLIST (see Chapter 5, *Environments*).

C.5.3 Terminating a BMP Extension

A successfully started BMP extension job will not end without intervention; the operator must terminate it in one of three ways:

- By executing job FOCBMPKX, described in this section. This is the recommended termination method.
- Through an MVS cancel.

The MVS cancel should be avoided because of its potential impact on the IMS/DC region if the cancellation occurs while the BMP extension is processing a DL/I call. The MVS cancel will also bypass the BMP extension's own cleanup process with the result that a 4K area in the CSA (Common Storage Area) will not be freed and will remain unavailable until the next IPL.

- Through an IMS cancel.

The IMS cancel will properly disconnect the BMP extension and IMS/DC regions without any impact on the latter, but it will bypass the freeing of the 4K communication area in the CSA.

Termination through job FOCBMPKX avoids problems. The job sends the BMP extension a termination message that breaks into the server message queue ahead of all other messages addressed to it. Upon receipt of this message, the BMP extension performs its cleanup and returns to IMS, signaling a normal completion.

Execute job FOCBMPKX to terminate a BMP extension. Sample JCL for this job follows

```
//FOCBMPKX EXEC PGM=FOCBMPKX,REGION=nn,PARM='ddname'
//STEPLIB DD DSN=prefix.IMS.LOAD,DISP=SHR
//ddname DD DSN=comfile,DISP=SHR
```

where:

nn Is the region size.

prefix Is the high-level qualifier for your site's FOCUS production libraries.

ddname Is allocated to the communication file of the BMP extension job to be canceled.

comfile Is the dataset name of the communication file that is allocated to the BMP extension you want to terminate; for example, 'prefix.FOCBMP.DATA.'

Each execution of the FOCBMPKX job terminates one BMP extension job.

Any of the three termination methods can be invoked at any time, without impact on those who might be actively using the canceled job (except for the impact on IMS/DC discussed previously). All such users will receive one of several error messages, depending on where in their retrieval process the cancellation occurred. Their retrieval ends cleanly; in no case will the user ABEND or be left with a hung terminal.

C.6 Accessing IMS Databases From CMS

For CMS access, you must install the FOCUS Cross-Machine Interface, as described in the *FOCUS for IBM Mainframe Cross-Machine Interface Users Manual*. Once installed, the Cross-Machine Interface performs the same function as the BMP extension. See Section C.5, *Accessing the BMP Extension*, for details.

D Installation Instructions

This appendix describes how to install the IMS/DB Interface. It includes instructions for installing the DBCTL environment and the XMI server. For information about these environments, see Chapter 5, *Environments*.

In prior releases, the BMP extension served the function now performed by the XMI server. In the current release, by default, users access the XMI server; for information about accessing the BMP extension, see Appendix C, *Release Dependent Interface Features*. The instructions for installing the XMI server also install the BMP extension.

D.1 Pre-Installation and Maintenance Requirements

Before you begin to install the Interface, you should be aware of installation prerequisites and consider maintenance procedures that may affect the installation process. This section describes pre-installation and maintenance requirements.

These instructions assume that the person performing the installation and maintenance procedures has a working knowledge of MVS. If you are installing the Interface with the FOCUS Multi-Session Option (MSO), MSO knowledge is required. Knowledge of FOCUS and IMS DL/I calls is not required.

Note: To access IMS databases from CMS, you must install the FOCUS Cross-Machine Interface as described in the *FOCUS for IBM Mainframe Cross-Machine Interface Users Manual*.

Your IMS database administrator will have to provide site-specific information.

Read this appendix thoroughly before installing the Interface to ensure correct installation.

D.1.1 Software Requirements

Before you install the Interface, please review the following list of software requirements:

- IMS must be installed and fully operational. If it is not, contact your IMS database administrator.
- FOCUS must be installed and fully operational. If it is not, contact your FOCUS database administrator or consult the appropriate FOCUS installation guide for installation instructions.

You also need to know the FOCUS Release and maintenance level. There are two ways to identify your release:

- Check the label of the distribution tape used to install FOCUS; the numbers are printed on it.
- Invoke FOCUS if it is already installed. The release is displayed each time you begin a session. To display the maintenance level as well, issue the ? RELEASE query command.

Every copy of FOCUS has a release number and a maintenance level.

D.1.2 Maintenance

There are no maintenance procedures that you must perform regularly to ensure proper functioning of the Interface. However, the following three situations require maintenance:

- If you install a new release of FOCUS (including maintenance releases), you must also reinstall the Interface from the same release.
- If you receive a program temporary fix (PTF) that affects the Interface, it will be accompanied by a cover letter containing installation instructions. If you still have installation questions after reading the cover letter, contact Information Builders Customer Support Services (CSS) in New York at (800) 736-6130 or your local Information Builders representative.
- If you install a new version of IMS, you do not have to re-install the Interface. However, you must make sure that the DFSRESLB DD card references the IMS.RESLIB dataset for the new version of IMS. For DBCTL, you must reassemble and re-link the DFSPZP module, as described in *Assemble and Link the DRA Startup Table* in Section D.3.3, *Create the DRA Startup Table: DFSPZPxx*.

D.2 Basic Installation Steps

The installation process starts with the following three steps regardless of which environments you use:

1. Allocate the Interface libraries (see Section D.2.1, *Allocate the Interface Libraries*).
2. Unload the distribution tape (see Section D.2.2, *Unload the Distribution Tape*).
3. Prepare the Interface run-time libraries (see Section D.2.3, *Prepare the Interface Run-time Libraries*).

The remaining installation steps depend on the environments you install. Section D.3, *DBCTL Instructions*, contains instructions for installing DBCTL, and Section D.4, *XMI Server Instructions*, contains instructions for installing the XMI server.

D.2.1 Allocate the Interface Libraries

The FOCUS distribution tape contains two partitioned datasets, or libraries, that support the Interface:

- IMS.LOAD, file 7 on the distribution tape, contains the load modules required to run the Interface.
- IMS.DATA, file 8 on the distribution tape, contains installation JCL and control statements for utility programs and for the linkage editor. This library can serve as an installation library, a maintenance library, and a run-time library.

The IMS.DATA PDS contains a sample Master File for the IMS DI21PART database. Before using it, you have to copy this file description, as described in Section D.2.3, *Prepare the Interface Run-time Libraries*, and you must create a corresponding FOCPSB.

Installation Instructions

Allocate space for these two libraries before unloading them. The following chart describes the allocation parameters and disk space requirements (in tracks) for each library:

Library	Descriptions	Disk Type
		3390
IMS.LOAD	Primary Allocation (tracks)	30
	Secondary Allocation	5
	Directory Blocks (BLKSIZE=13030, RECFM=U,DSORG=PO)	20
IMS.DATA	Primary Allocation (tracks)	20
	Secondary Allocation	5
	Directory Blocks (LRECL=80, BLKSIZE=1600, RECFM=FB, DSORG=PO)	25

Depending on the device used at your site, your allocations may differ from those in the table.

Note: The high-level qualifier you specify for each library should be the same as the high-level qualifier used for other FOCUS datasets at your site. The examples throughout this appendix use the identifier 'prefix' to refer to this high-level qualifier.

D.2.2 Unload the Distribution Tape

Use the following sample JCL to unload the contents of the tape. Add a JOB card, VOL=SER values, and UNIT values. The VOL=SER values are provided on the external tape label.

Make any changes needed to conform to your site's standards, and then submit the JCL for execution.

```
//UNLOAD EXEC PGM=IEBCOPY
//LOADIN DD DSN=IMS.LOAD,DISP=SHR,
// UNIT=unittype,VOL=SER=valid,
// LABEL=(7,SL,EXPDT=98000)
//DATAIN DD DSN=IMS.DATA,DISP=SHR,
// UNIT=unittype,VOL=SER=valid,
// LABEL=(8,SL,EXPDT=98000)
//LOADOUT DD DSN=prefix.IMS.LOAD,DISP=OLD
//DATAOUT DD DSN=prefix.IMS.DATA,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
COPY INDD=LOADIN,OUTDD=LOADOUT
COPY INDD=DATAIN,OUTDD=DATAOUT
/*
```

where:

<i>unittype</i>	Is the unit type on which a dataset resides, such as TAPE or CART.
<i>valid</i>	Is the volume identifier (serial number) of a direct access volume to contain an Interface library.
<i>prefix</i>	Is the high-level qualifier for your site's Interface libraries.

D.2.3 Prepare the Interface Run-time Libraries

At this point, you may need to create or modify seven run-time libraries. At execution time, users allocate these libraries with a CLIST, a FOCEXEC, or with batch JCL, as described in Chapter 5, *Environments*.

Many sites create private and public versions of the FOCEXEC, MASTER, FOCPSB, and ACCESS libraries to make maintenance easier. For example, you can create private FOCEXEC libraries for users' personal sets of FOCEXECs and retain one general site FOCEXEC library for all users to access.

If you want private and public libraries, you can create the additional libraries now. (First check to see if private libraries already exist.) The additional libraries should have the same DCB attributes as the original libraries.

Each Interface user must have access to the libraries described in the following sections.

The MASTER Library

The MASTER library was created when FOCUS was installed. The library is allocated to ddname MASTER and its members are Master Files.

A Master File describes the segments and fields in an IMS database. Each database accessed by the Interface must be described to FOCUS in a Master File.

Before running any of the examples from this manual, copy the following member from 'prefix.IMS.DATA' to 'prefix.MASTER.DATA':

Member Name	Description
DI21PART	Describes the IMS DI21PART database.

The FOCPSB Library

You must create 'prefix.FOCPSB.DATA,' the FOCPSB library; at execution time it will be allocated to ddname FOCPSB. Its members are FOCPSBs. A FOCPSB describes an IMS PSB.

The following example illustrates how to create the FOCPSB library using the IEFBR14 utility

```
//STEP1 EXEC PGM=IEFBR14
//FOCPSB DD DSN=prefix.FOCPSB.DATA,DISP=(NEW,CATLG),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600),
//          SPACE=(TRK,(p,s,d)),VOL=SER=xxx
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

p Is the primary space allocation.

s Is the secondary space allocation.

d Is the allocation for directory blocks.

xxx Is a valid volume id for your site.

Before running any of the examples from this manual, create a member in 'prefix.FOCPSB.DATA' to correspond to the PSB for the DI21PART database. Chapter 3, *Creating FOCUS Descriptions*, discusses FOCPSBs and Appendix A, *Sample File Descriptions*, illustrates a sample PSB and FOCPSB for the DI21PART Master File.

The ACCESS Library

If you install the DBCTL environment (see Section D.3, *DBCTL Instructions*), you should create 'prefix.ACCESS.DATA,' the ACCESS library; at execution time it will be allocated to ddname ACCESS. Its members are Access Files. Access Files are optional in the IMS/DB Interface; they can be used to select the PSB for a request. Each Access File has a corresponding Master File in the MASTER library.

The following example illustrates how to create the ACCESS library using the IEFBR14 utility

```
//STEP1 EXEC PGM=IEFBR14
//ACCESS DD DSN=prefix.ACCESS.DATA,DISP=(NEW,CATLG),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600),
//          SPACE=(TRK,(p,s,d)),VOL=SER=xxx
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

p Is the primary space allocation.

s Is the secondary space allocation.

d Is the allocation for directory blocks.

xxx Is a valid volume id for your site.

The FOCEXEC Library

The FOCEXEC library was created when FOCUS was installed. The library was allocated to ddname FOCEXEC and its members are FOCUS procedures (FOCEXECs).

The Interface Load Library

The Interface load library ('prefix.IMS.LOAD') was unloaded from the distribution tape. This library contains essential Interface software, including the Interface load module, member IMSX.

There are three installation alternatives for allocating the Interface load library. At execution time, FOCUS searches for the Interface modules first in the datasets allocated to ddname USERLIB; if USERLIB is not allocated or does not contain the modules, FOCUS searches the datasets allocated to ddname STEPLIB. You can therefore do any of the following:

- Concatenate 'prefix.IMS.LOAD' after the FOCUS load library ('prefix.FOCLIB.LOAD') in the allocation for ddname STEPLIB:

```
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//          DD DSN=prefix.IMS.LOAD,DISP=SHR
```

- Copy the Interface modules into 'prefix.FOCLIB.LOAD,' and allocate this combined load library as ddname STEPLIB:

```
COPY 'prefix.IMS.LOAD(*)' 'prefix.FOCLIB.LOAD(*)' NONUM
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
```

- Allocate 'prefix.IMS.LOAD' as ddname USERLIB:

```
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//USERLIB DD DSN=prefix.IMS.LOAD,DISP=SHR
```

All options are valid for both TSO and batch access. However, under TSO, ddname STEPLIB cannot be allocated dynamically with the ALLOCATE command; therefore, if it is to be used, it must be allocated in the logon procedure. All three options produce equivalent execution time overhead, but USERLIB requires 26K bytes of extra storage for I/O buffers.

The ERRORS Library

The ERRORS library ('prefix.ERRORS.DATA') was created when FOCUS was installed. It contains the text of the FOCUS error messages. Member FOCIMS of 'prefix.IMS.DATA' contains the text of all the error messages generated by the IMS/DB Interface. You must copy this member to the ERRORS library:

```
COPY 'prefix.IMS.DATA(FOCIMS)' 'prefix.ERRORS.DATA(FOCIMS)' NONUM
```

At execution time, allocate the receiving dataset as ddname ERRORS:

```
ALLOCATE F(ERRORS) DA('prefix.ERRORS.DATA') SHR REUSE
```

The file of messages is delivered without line numbers; this feature is preserved through the NONUM option of the TSO COPY command.

The Maintenance Library

Copy member IMSX from 'prefix.IMS.DATA' to member IMSX of 'prefix.FOCCTL.DATA.' This file is used to apply PTF Maintenance to module IMSX.

D.3 DBCTL Instructions

You can install the Interface to use the DBCTL subsystem of IMS if your site is running IMS version 3.1 or higher. Execute the following installation steps:

1. Test your IMS DBCTL installation (see Section D.3.1, *Test Your DBCTL Installation: IMDTST*).
2. Modify APPLCTN macros (see Section D.3.2, *Modify APPLCTN Macros*).
3. Create the DRA Startup Table and place it in the 'prefix.IMS.LOAD' library. Concatenate this library ahead of IMS.RESLIB in the allocation for ddname STEPLIB (see Section D.3.3, *Create the DRA Startup Table: DFSPZPxx*).
4. If you use MSO, select DBCTL mode by including the attribute DBCTL=START in the configuration file (see Section D.3.4, *Create an MSO Configuration File and JCL for the DBCTL Environment*).

Instructions for invoking DBCTL once it is installed are in Chapter 5, *Environments*.

D.3.1 Test Your DBCTL Installation: IMDTST

The IMDTST JCL, provided in the 'prefix.IMS.DATA' library, executes program IMDTEST to verify that DBCTL is properly installed. Run it before installing the IMS/DB Interface.

Note: If you are using IMS/ESA V3.1, make sure you have applied APAR PL70841.

The following is sample JCL for executing program IMDTEST

```
//job card goes here
//STEP1      EXEC PGM=IMDTEST,PARM='emppsbl,pcicspsb,xx',
//           REGION=4096K,TIME=(1,5)
//STEPLIB    DD DSN=prefix.IMS.LOAD,DISP=SHR
//           DD DSN=IMS.RESLIB,DISP=SHR
//SYSIN      DD DUMMY
//SYSOUT     DD SYSOUT=*
//SYSPRINT   DD SYSOUT=*
```

where:

emppsbl Is the name of the PSB to schedule.

pcicspsb Is a security class.

xx Is a 2-character suffix that identifies the DRA Startup Module to be loaded, as described in Section D.3.3, *Create the DRA Startup Table: DFSPZPxx*. **Note:** If you omit this parameter, its value defaults to 00, which may cause IMS to load the incorrect module.

prefix Is the high-level qualifier for your site's FOCUS production libraries.

After running the JCL, examine the JES output; the return codes displayed on the following lines indicate that DBCTL is properly installed:

```
JOB05736 +CCTL: JOBNAME(USERID1) PSB(EMPPSB1 ) CLASS(PCICSPSB) SUFFIX(5F)
JOB05736 +CCTL: INIT CALLED
JOB05736 +CCTL: INIT RETURNED,RETC(00000000)
JOB05736 +CCTL: WAITING FOR CONTROL EXIT
JOB05736 +CTLX: CONTROL EXIT CALLED
JOB05736 +CTLX: CONTROL EXIT RETURNED
JOB05736 +CCTL: INIT COMPLETED, FUNC(02) SFNC(00) RCOD(00) RETC(00000000)
JOB05736 +CCTL: SECURITY CHECK COMPLETED, RC(00000004)
JOB05736 +CCTL: TERMINATE DRA CALLED
JOB05736 +CTLS: SUSPEND EXIT CALLED
JOB05736 +CTLR: RESUME EXIT CALLED
JOB05736 +CTLS: SUSPEND EXIT RETURNED
JOB05736 +CTLR: RESUME EXIT RETURNED
JOB05736 +CCTL: TERMINATE DRA COMPLETED, RETC(00000000)
```

Note that the installation is successful even when the security check completes with a return code of 4.

If the JES output indicates an error, make sure of the following criteria:

- A DRA Startup Table with the suffix you indicated on the EXEC card exists. If it exists, check the following things:
 - The parameters specified in it (especially the DSNAME and DDNAME parameters) are correct.
 - The DBCTLID keyword points to the correct IMS system.
- DBCTL was installed at your site.

If DBCTL was installed, check that it is up and running.

See Section D.3.3, *Create the DRA Startup Table: DFSPZPxx*, for information about the DRA Startup Table.

D.3.2 Modify APPLCTN Macros

Make sure all APPLCTN macros describing PSBs that will be accessed by multiple users specify SCHDTYP = Parallel. If necessary, modify the APPLCTN macros for such PSBs to include the attribute SCHDTYP = Parallel and re-run the IMS SYSGEN to make the changes effective.

D.3.3 Create the DRA Startup Table: DFSPZPxx

The Database Resource Adapter (DRA) is the interface between a user task and DBCTL. The DRA Startup Table contains values that define the characteristics of the DRA. The DRA Startup Table is named DFSPZPxx, where xx is a two-character suffix that should be chosen to comply with your site's standards.

Choose the Suffix for the DRA Startup Table

You must choose a suffix for the DRA Startup Table that complies with your site's standards. Once this suffix has been chosen, user applications must identify the chosen suffix to the Interface:

- MSO applications identify the chosen suffix by including the IMSPZP attribute in the DBCTL definition of the configuration file. Section D.3.4, *Create an MSO Configuration File and JCL for the DBCTL Environment*, includes an example.
- TSO applications identify the chosen suffix by issuing a SET command, as described in Chapter 5, *Environments*.

Assemble and Link the DRA Startup Table

Create the DRA Startup Table by specifying parameters in the DFSPRP macro of the DFSPZPxx module and then assembling and linking the DFSPZPxx module into the 'prefix.IMS.LOAD' library, as illustrated in the following sample JCL.

The sample JCL illustrates how to specify the parameters. You can normally find this JCL in your IMS installation library. A chart of the most common keywords and their descriptions follows the sample JCL.

```
//job card goes here
//ASSEMBLE EXEC PGM=IEV90,REGION=2M,PARM='OBJECT,NODECK'
//SYSLIB DD DSN=IMS.MACLIB,DISP=SHR
//SYSLIN DD UNIT=SYSDA,DISP=(,PASS),
//          SPACE=(80,(100,100),RLSE),
//          DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1 DD UNIT=SYSDA,DISP=(,DELETE),
//          SPACE=(CYL,(10,5))
//SYSIN DD *
PRP TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZPxx CSECT
EJECT
DFSPRP DSECT=NO, X
        DBCTLID=IMS3, X
        DDNAME=DFSRESLB, X
        DSNAME=IMS.RESLIB, X
        CNBA=150, X
        MAXTHRD=150, X
        MINTHRD=5
END

/*
/**
//LINK EXEC PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,ASSEMBLE),
//          REGION=4M
//SYSLIN DD DSN=*.ASSEMBLE.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//          SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
//SYSLMOD DD DISP=SHR,DSN=prefix.IMS.LOAD(DFSPZPxx)
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

xx Is the two-character suffix chosen for the DRA Startup Table.

Note: The 'prefix.IMS.LOAD' library must be concatenated ahead of IMS.RESLIB in the allocation for ddname STEPLIB.

Installation Instructions

The following chart describes the most common keywords as defined for IMS/ESA Version 4. Other parameters that affect your IMS environment and that may be required at your site are described in the *IBM IMS/ESA Version 4 System Definition Reference*.

Keyword	Description	Default
AGN	A one- to eight-character application group name used as part of the DBCTL security function. For more information on DBCTL security, see the <i>IMS/ESA System Administration Guide</i> .	N/A
CNBA	The total number of Fast Path buffers for the MSO server's use. For a description of Fast Path DEDB buffer usage, see the <i>IMS/ESA System Administration Guide</i> . You can omit this parameter if your site does not utilize Fast Path.	N/A
DBCTLID	The four-character name of the DBCTL region. This is the same as the IMSID parameter in the DBC procedure. For more information on the DBC procedure, see the <i>IBM IMS/ESA Version 3 System Definition Reference</i> .	SYS1
DDNAME	Must be DFSRESLB. It is the ddname that will be allocated to the DBCTL RESLIB library (see the DSNAME keyword).	N/A
DSNAME	The one- to 44-character data set name of the DBCTL RESLIB library; it must contain the DRA modules and be MVS authorized.	IMS.RESLIB
FPBOF	The number of Fast Path DEDB overflow buffers to be allocated per thread. For a description of Fast Path DEDB buffer usage, see the <i>IMS/ESA System Administration Guide</i> . You can omit this parameter if your site does not utilize Fast Path.	00
FPBUF	The number of Fast Path DEDB buffers to be allocated and fixed per thread. For a description of Fast Path DEDB buffer usage, see the <i>IMS/ESA System Administration Guide</i> . You can omit this parameter if your site does not utilize Fast Path.	00

Keyword	Description	Default
FUNCLV	The level of the DRA that the CCTL supports. FUNCLV=1 means the CCTL uses the DRA at the IMS 3.1 level.	1
MAXTHRD	The maximum number of concurrent DRA threads to be available. The maximum is 255.	1
MINTHRD	The minimum number of concurrent DRA threads to be available. Since the number of threads specified by this parameter will be allocated at all times, regardless of whether they are used, be careful in selecting this value. The maximum is 255.	1
SOD	The output class to use for a SNAP DUMP of abnormal thread termination.	A
TIMEOUT	The number of seconds a CCTL should wait for the successful completion of a DRA TERM request. Specify this value only if the CCTL is coded to use it. This value is returned to the CCTL upon completion of an INIT request.	60
TIMER	The number of seconds between attempts of the DRA to identify itself to DBCTL during an INIT request.	60
USERID	An eight character name of the CCTL region. No two CCTLs (MSO servers accessing the same IMS region through DBCTL) can have the same USERID (address space ID).	N/A

D.3.4 Create an MSO Configuration File and JCL for the DBCTL Environment

From TSO, you select the DBCTL environment with a SET command (see Chapter 5, *Environments*). To select the DBCTL environment from MSO, you must include the attribute DBCTL=START in the configuration file. This section illustrates a sample configuration file, sample MSO JCL, and instructions for controlling the DBCTL environment with console commands.

Sample Configuration File

The following sample configuration file includes the attributes needed for invoking DBCTL. Your system administrator can supply you with configuration file values, including IMSCLASS, that are specific to your site.

The MSO configuration file is allocated to ddname FOCMSO in the MSO server JCL

```
NUMFOC=1
MAXFOC=8
MSOLOG=ON
EXTSEC=ON
ATTNKEY=PA2
LU2_NAME=CICSE
*****
** D B C T L   K E Y W O R D S               **
*****
IMSPZP=xx
DBCTL=START
IMSSEC=ON
IMSCLASS=PCICSPB
*****
** S E R V I C E == F O C U S               **
*****
SERVICE=FOCUS
PROGRAM=FOCUS
NUMBER_READY=0
MAXIMUM=30
```

where:

xx Is the two-character suffix chosen for the DRA Startup Table, described in Section D.3.3, *Create the DRA Startup Table: DFSPZPxx*. If omitted, it defaults to 00.

Note: You must place the IMSPZP attribute before the DBCTL attribute.

Sample JCL

Use the following MSO server JCL as a model

```
//MSO      EXEC  PGM=SSCTL
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR
//         DD DSN=prefix.FUSELIB.LOAD,DISP=SHR
//         DD DSN=prefix.IMS.LOAD,DISP=SHR
//         DD DSN=IMS.RESLIB,DISP=SHR
//ERRORS   DD DSN=prefix.ERRORS.DATA,DISP=SHR
//         DD DSN=prefix.IMS.DATA,DISP=SHR
//MSGGET   DD DSN=prefix.MSO.MSOGET,DISP=SHR
//MSPUT    DD DSN=prefix.MSO.MSOPUT,DISP=SHR
//MSOPROF  DD DSN=prefix.MSOPROF.DATA,DISP=SHR  *MSO Profile FOCEXEC
//FOCSERVE DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//FOCEXEC  DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
//FOCMSO   DD DSN=prefix.MSO.MSOCONFIG,DISP=SHR *MSO config file
//MASTER   DD DSN=prefix.MASTER.DATA,DISP=SHR
//ACCESS   DD DSN=prefix.ACCESS.DATA,DISP=SHR
//FOCPSB   DD DSN=prefix.FOCPSB.DATA,DISP=SHR
//MSOPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

Note: In the sample, ddname MSOPROF is allocated to the file that contains your MSO profile FOCEXEC. If you store this profile as a member of your '*prefix*.FOCEXEC.DATA' library, allocate ddname MSOPROF to that library.

If you use other FOCUS Interfaces in addition to the IMS Interface, you must allocate additional datasets (such as load libraries, Access File libraries, and error datasets) specific to those Interfaces. For more information, see the *FOCUS for IBM Mainframe Multi-Session Option Installation and Technical Reference Guide*.

To connect to the DBCTL environment after the MSO server is running, see the next section.

Console Commands for Controlling the DBCTL Environment

The system administrator can use the MSO or MVS operator's console to start and stop the DBCTL thread, modify the security class, and change the DFSPZPxx module that is loaded at initialization time.

The following commands can be issued from the console

```
/[F jobname,]IMSPZP=xx
/[F jobname,]DBCTL= START
/[F jobname,]DBCTL= STOP[,I]
/[F jobname,]IMSSEC= {ON|OFF}
/[F jobname,]IMSCLASS=classname
/[F jobname,]DBCTL ?
```

where:

/	Is required syntax.
F jobname,	Is syntax for issuing a command from the MVS operator's console.
xx	Sets the name of the DRA Startup Table (see Section D.3.3, <i>Create the DRA Startup Table: DFSPZPxx</i>). Must be issued prior to the /DBCTL=START command.
START	Starts the DBCTL connection to IMS.
STOP	Stops the DBCTL connection to IMS after waiting for all currently-executing requests to finish.
STOP,I	Stops the DBCTL connection to IMS immediately; does not wait for requests to finish.
ON	Enables security checking as described in Chapter 6, <i>Security</i> .
OFF	Disables security checking as described in Chapter 6, <i>Security</i> .
classname	Is a valid class that contains security authorization information. See Chapter 6, <i>Security</i> .
?	Displays current DBCTL settings.

Note: If you issue the DISPLAY USER (DU) command from the MSO console, a new column (PSB) displays the name of the active PSB.

For more information about the MSO console, consult the *FOCUS for IBM Mainframe Multi-Session Option Installation and Technical Reference Guide*.

Shutting Down IMS in the DBCTL Environment

If the system operator shuts IMS down, the Interface terminates all active DBCTL threads immediately.

The proper way to shut down DBCTL consists of the following steps:

1. Determine whether any active requests are running.
 - If no active requests are running, issue the `/DBCTL STOP` console command.
 - If active requests are running, issue the `/DBCTL STOP,I` console command.
2. If you are running MSO, shut down the MSO server. If you are in TSO, exit from FOCUS.

D.4 XMI Server Instructions

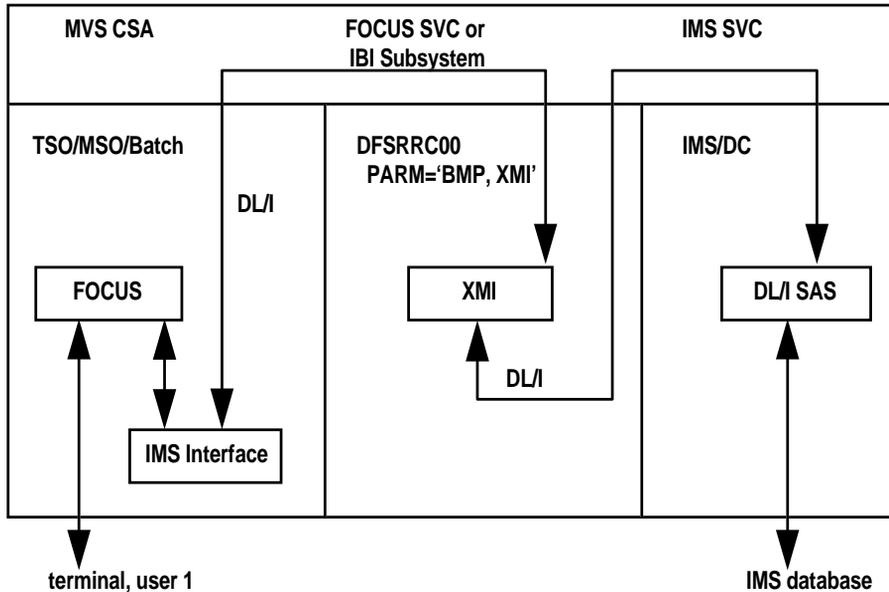
In the XMI server environment, FOCUS runs in an address space that contains no IMS components.

In prior releases, the BMP extension served the function now performed by the XMI server. In the current release, by default, users access the XMI server; for information about accessing the BMP extension, see Appendix C, *Release Dependent Interface Features*. The instructions for installing the XMI server also install the BMP extension.

The XMI server uses either a Supervisor Call (SVC) or the IBI Subsystem to communicate between the user ID and the central database manager that runs as a separate job and performs all the IMS database I/O. For information about the SVC or the IBI Subsystem, see the *FOCUS for IBM Mainframe MVS/TSO Installation Guide*.

Installation Instructions

The following diagram illustrates the relationship between the XMI server, FOCUS, and the central database manager:



The MVS System Support Group should have installed the SVC or the IBI Subsystem when FOCUS was installed. No additional SVC installation is required.

If neither the SVC nor the IBI subsystem was installed during FOCUS installation, install one of them now according to the instructions in the *FOCUS for IBM Mainframe MVS/TSO Installation Guide*.

Execute the following steps to install the XMI server:

1. Allocate communication files (see Section D.4.1, *Allocate Communication Files*).
2. Create JCL for XMI server jobs (see Section D.4.2, *Create JCL for XMI Server Jobs*).

D.4.1 Allocate Communication Files

The XMI server runs in a different address space from FOCUS and the IMS/DB Interface. The two address spaces must establish a link before they can communicate with one another. A communication file assists this handshake function by acting as a bulletin board, notifying FOCUS about the XMI server's whereabouts. After the handshake is completed, the communication file plays no role in subsequent message traffic.

You can initiate up to 16 concurrent XMI server jobs, each of which requires its own communication file.

You must now allocate as many communication files as needed, up to sixteen. Dataset names are irrelevant but, for clarity, you can use the names 'prefix.FOCBMP.DATA,' 'prefix.FOCBMP1.DATA,' ... 'prefix.FOCBMP15.DATA' (these names correspond to the ddnames that users allocate when invoking the Interface). You can make the initial allocations through the IEFBR14 utility or the TSO ALLOCATE command, provided that it can catalog datasets on permanently mounted disk packs.

The following example illustrates the allocation of two communication files using the IEFBR14 utility

```
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=prefix.FOCBMP.DATA,DISP=(NEW,CATLG),DCB=BLKSIZE=16,
// SPACE=(TRK,(1)),VOL=SER=xxx
//DD2 DD DSN=prefix.FOCBMP1.DATA,DISP=(NEW,CATLG),DCB=BLKSIZE=16,
// SPACE=(TRK,(1)),VOL=SER=xxx
```

where:

prefix Is the high-level qualifier for your site's FOCUS production datasets.

xxx Is a valid volume id for your site.

Note that the minimum possible amount of space is allocated. Users must have read access to the communication files.

At execution time, each XMI server job allocates a *different* communication file to ddname FOCBMP:

```
//FOCBMP DD DSN=prefix.FOCBMP1.DATA,DISP=SHR
```

In order for all users to execute a common CLIST, each user program accessing the XMI server should allocate *all* communication files, to ddnames FOCBMP, and FOCBMP1 through FOCBMP15:

```
ALLOC F(FOCBMP) DA('prefix.FOCBMP.DATA') SHR
ALLOC F(FOCBMP1) DA('prefix.FOCBMP1.DATA') SHR
.
.
.
ALLOC F(FOCBMP15) DA('prefix.FOCBMP15.DATA') SHR
```

You can use any of the sixteen reserved ddnames, and you can pair them with any of the dataset names. However, all of the communication files must be allocated in SHR mode, both by the user and by the XMI server job.

Note: When running MSO, it is recommended that you allocate all communication files in the MSO server JCL.

D.4.2 Create JCL for XMI Server Jobs

The JCL to use for initiating an XMI server job varies depending on whether IMS or CICS has control of the databases and whether the databases are online or local. In addition, XMI server jobs differ in two ways:

- The PSB name, which is user-selected and is passed to the job as a parameter.
- The communication file, which is a different file in each XMI server job but is always allocated to ddname FOCBMP.

The job class and time expiration parameters used with the XMI server job must allow for a long residence in the system, and the priority should be sufficiently high to give good response to the users being serviced. In setting the various job parameters that affect response, remember that these jobs spend most of their time in the WAIT state consuming no resources, since they have nothing to do when not responding to DL/I calls. Even when processing such calls, they only do minimal work. All the logical work of the Interface is done in the various TSO or MSO address spaces.

To prepare JCL for an XMI server job, consult Chapter 5, *Environments*; it describes the various options available and provides JCL for each option. The following is one example

```
//XMIBMP EXEC PGM=DFSRRC00,PARM='BMP,XMI,psbname'  
//STEPLIB DD DSN=prefix.FOCLIB.LOAD,DISP=SHR  
// DD DSN=prefix.IMS.LOAD,DISP=SHR  
// DD DSN=IMS.RESLIB,DISP=SHR  
//FOCBMP DD DSN=prefix.FOCBMP.DATA,DISP=SHR  
//FOCPSB DD DSN=prefix.FOCPSB.DATA(psbname),DISP=SHR  
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR  
//ERRORS DD DSN=prefix.ERRORS.DATA,DISP=SHR  
//SYSPRINT DD SYSOUT=A
```

where:

prefix Is the high-level qualifier for your site's FOCUS production libraries.

psbname Is the member of the FOCPSB dataset that corresponds to the PSB to use.

Note:

- STEPLIB must allocate module XMI and the usual IMS execution time modules.
- Ddname FOCBMP is allocated to one of the communication files discussed in Section D.4.1, *Allocate Communication Files*. It is essential to specify DISP=SHR because this file is also allocated by users when they invoke the Interface.
- Ddname FOCPSB is allocated to the member of the FOCPSB dataset that corresponds to the PSB selected.
- Ddname ERRORS points to the PDS of FOCUS error messages that contains member FOCIMS. This dataset must be allocated both by the user and by the XMI server job.
- There is no SYSIN file.
- If this job is submitted for databases under control of CICS, you must allocate ddname DFHLIB to the CICS.LOAD library containing the member DFHDRP (see Chapter 5, *Environments*, for changes to the EXEC card). If your site uses authorized libraries, verify that all FOCUS libraries have the required authorizations.
- Most error conditions detected by the XMI server are reflected back to the originating user, where they are issued to the terminal. The SYSPRINT file allocated in the XMI server job will contain only global error messages pertaining to the XMI server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.

Note: Instructions for terminating an XMI server job are in Chapter 5, *Environments*.

D.4.3 Shutting Down IMS in the XMI Server Environment

Prior to bringing IMS down, you must run job XMIKILL (described in Chapter 5, *Environments*) to cancel any XMI servers that were initiated with the parameter BMP.

D.5 Run-time Requirements

The Interface now resides on your system. Before you invoke the Interface, you must do two things:

- Create a program (a CLIST or REXX EXEC for TSO, a FOCEXEC for MSO, or JCL for a batch environment) that invokes FOCUS, with the Interface, from each environment you installed. Chapter 5, *Environments*, includes sample CLISTs and JCL.
- Invoke FOCUS and run a simple query to test the Interface installation.

The following sample request uses the DI21PART Master File that is supplied with the Interface. Before submitting this request, make sure you created a FOCPSB for the PSB you use with the DI21PART database. FOCPSBs are discussed in Chapter 3, *Creating FOCUS Descriptions*. Appendix A, *Sample File Descriptions*, includes examples of a PSB and a FOCPSB for DI21PART:

```
TABLE FILE DI21PART
PRINT PARTKEY
IF RECORDLIMIT EQ 10
END
```

E Interface Errors and Messages

This appendix contains a list of Interface error and informational messages (see Section E.1, *Interface Messages*) and a discussion of the most common errors (see Section E.2, *Common User Errors*).

E.1 Interface Messages

The following is a list of Interface messages. They are stored in 'prefix.IMS.DATA(FOCIMS)' for MVS and in FOCIMS ERRORS for CMS, and they are subject to change.

Messages that refer to the XMI server also apply to the BMP extension.

On-line explanations are available for some messages. To see the explanatory information, issue the following from the FOCUS command level

? n

where:

n Is a message number.

- (FOC4201) **RETRIEVAL REQUEST DOES NOT REFERENCE ANY DATABASE FIELDS**
The user's request does not reference any database fields. For example, it uses only defined fields that require no access to data, such as $X = X + 1$.
- (FOC4202) **INSUFFICIENT MEMORY FOR IMS INTERFACE**
More memory is needed to construct tables and SSAs. Generally about 4K of additional memory will suffice. If that is not possible, reduce the sort work area by reducing the number of bins through the command SET BINS = m. A minimum of 13 bins is required.
- (FOC4203) **LOGIC ERROR IN SSA GENERATION**
Interface logic error—call your IBI representative.
- (FOC4204) **NO LOCAL PCB FOR FILENAME**
No PCB could be found corresponding to the displayed Master File member name. Verify that the Master File member name has a matching entry in the FOCPSB description.
- (FOC4205) **CANNOT LINK TO XMI REGION FOR DDNAME**
The displayed ddname allocated a communication dataset but there is no active XMI server job for it.
- (FOC4206) **CANNOT ENQUEUE ON MAILBOX OF XMI REGION FOR DDNAME**
Communication error—call your IBI representative.

Interface Errors and Messages

- (FOC4207) CANNOT TRANSMIT PCB MESSAGE TO XMI REGION FOR DDNAME
Communication error—call your IBI representative.
- (FOC4208) CANNOT RECEIVE PCB REPLY FROM XMI REGION FOR DDNAME
Communication error—call your IBI representative.
- (FOC4209) CANNOT DEQUEUE MAILBOX OF XMI REGION FOR DDNAME
Communication error—call your IBI representative.
- (FOC4210) BUSY PCB RETURNED BY XMI REGION FOR DDNAME
Communication error—call your IBI representative.
- (FOC4211) NO XMI REGION HAS ANY PCB FOR FILENAME
The XMI server is used, but none of the active XMI server jobs has a PCB corresponding to the displayed filename. The FOCPSB allocation may have been omitted from the server JCL.
- (FOC4212) NO FREE PCB IN ANY XMI REGION FOR FILENAME
The XMI server is used and all PCBs corresponding to the displayed filename are in use. Try again later.
- (FOC4213) LOCAL DLI CALL ERROR STATUS FOR SEGMENT
The XMI server is not used and the locally issued DL/I call returned the displayed IMS error code. Refer to Section E.2, *Common User Errors*, for typical STATUS codes that accompany this message.
- (FOC4214) REMOTE DLI CALL ERROR STATUS FOR SEGMENT
The XMI server is used and the XMI server job servicing this retrieval obtained the displayed IMS error message code. Refer to Section E.2, *Common User Errors*, for typical STATUS code values that accompany this message.
- (FOC4215) LOCAL PSB HAS MORE THAN ONE PCB FOR FILENAME
- (FOC4216) XMI PSB HAS MORE THAN ONE PCB FOR FILENAME
- (FOC4217) ERROR CREATING BMP MESSAGE FOR SEGMENT
Communication error—call your IBI representative.
- (FOC4218) BMP MESSAGE TOO LONG FOR SEGMENT
Communication error—call your IBI representative.
- (FOC4219) CANNOT ENQ MAILBOX .. MAILBOX STILL IN USE
Communication error—call your IBI representative.
- (FOC4220) CANNOT SEND MESSAGE .. MAILBOX STILL IN USE
Communication error—call your IBI representative.
- (FOC4221) CANNOT RECEIVE MESSAGE .. MAILBOX STILL IN USE
Communication error—call your IBI representative.

- (FOC4222) **I/O ERROR ON FILE FOCPSB**
File FOCPSB was allocated and an I/O error occurred trying to read it.
- (FOC4223) **INVALID PCB ATTRIBUTE FOR FILENAME**
File FOCPSB is allocated and contains a record with an invalid PCBTYPE attribute.
- (FOC4224) **FILE FOCPSB IS ALLOCATED BUT IT IS EMPTY**
File FOCPSB was allocated but was empty or contained only comment records.
- (FOC4225) **JOIN MUST BE DONE TO THE ROOT SEGMENT**
The 'TO' field in the JOIN must be in the root segment as seen through the PCB. Check the PSBGEN and clear the JOIN with the following command:
 JOIN CLEAR joinname
- (FOC4226) **'TO' FIELD IN JOIN DOESN'T HAVE IMS KEY**
The 'TO' field must have an ALIAS value that includes the suffix .KEY or .HKY in the Master File of the database cross-referenced in the JOIN (e.g., ALIAS=SSN.KEY).
- (FOC4227) **ACTUAL AND USAGE FORMATS OF 'FROM' & 'TO' FIELDS DIFFER**
The actual format of the 'FROM' and the usage format of the 'TO' field must be the same. No format conversion is allowed or will be performed.
- (FOC4228) **ACTUAL AND USAGE FORMATS OF 'TO' FIELD ARE DIFFERENT**
The actual and usage formats of the 'TO' field must be the same. No data conversion is allowed or needed.
- (FOC4229) **FORMATS OF 'TO' AND 'FROM' FIELDS ARE DIFFERENT**
The formats of 'TO' and 'FROM' fields must be the same or their values must be comparable.
- (FOC4230) **GROUP FLDS IN JOIN MUST HAVE AN EQUAL NUMBER OF ELEMENTS**
- (FOC4231)
- (FOC4232) **CALLIMS ERROR ON ISRT CALL**
- (FOC4233) **CALLIMS ERROR ON PURG CALL**
- (FOC4234) **'TO' SEGMENT IN JOIN IS NOT S1,S2,SH1,SH2**
The 'TO' field is in a segment defined with a SEGTYPE of S0 or blank, indicating that there is no key for the segment. The JOIN command supports only keyed IMS segments—SEGTYPEs S1, S2, SH1, and SH2.
- (FOC4235) **WARNING: DATA RETRIEVAL TERMINATED DUE TO DLI LIMIT**
Data retrieval was terminated because of a limit imposed by the DBA. A report is displayed but may be incomplete due to this resource restriction.

Interface Errors and Messages

- (FOC4236) **GATEWAY MACHINE IS NOT RUNNING**
Unable to communicate with the gateway machine. The gateway machine must be operating prior to a request for data from a DL/I database. Gateway user ID is not logged on or the gateway program is not running.
- (FOC4237) **GATEWAY - TARGET LINK IS NOT OPERATIONAL**
The gateway machine was unable to communicate with the database server in the other operating environment. Most likely the database server was not started prior to the report request.
- (FOC4238) **VMCF ERROR COMMUNICATING TO THE GATEWAY MACHINE**
Communication error. Most likely the gateway machine was logged off in the middle of a request. Restart the gateway and rerun the report request.
- (FOC4239) **CALLIMS ERROR ON CHKP CALL**
- (FOC4240) **CALLIMS ERROR ON GN CALL**
- (FOC4241) **BADPARMS TO CALLIMS**
- (FOC4242) **USE COMMAND IS MISSING FOR FILE**
The USE command for access to a DL/I database does not include the filename displayed in the error message.
- (FOC4243) **GATEWAY ID NAME IS MISSING IN THE USE COMMAND FOR FILE**
The gateway user ID is missing from the USE command. The USE command syntax to access DL/I data is either of the following:

 USE FILENAME DLI * ON GATEUSERID
 USE * DLI * ON GATEUSERID
- (FOC4244) **JOIN CAN BE DONE ONLY TO A FULL HDAM KEY**
If the 'TO' database is HDAM, you cannot do a short-to-long (partial to full) key JOIN.
- (FOC4245) **Reserved**
This message is reserved for CALLIMS client-server communication.
- (FOC4246) **Reserved**
This message is reserved for CALLIMS client-server communication.
- (FOC4247) **INVALID IMS COMMAND**
- (FOC4248) **INVALID PARAMETER IN IMS SET COMMAND**
- (FOC4249) **PSB NAME IS MISSING**
- (FOC4250) **DBCTL IS NOT INITIALIZED FOR THIS ADDRESS SPACE**
- (FOC4251) **DBCTL INITIALIZATION FAILED WITH**
- (FOC4252) **IMD IS OUT OF MEMORY**

(FOC4253) BAD PARAMETER LIST ON IMD CALL, RC

(FOC4254) ERROR LOADING DRA STARTUP MODULE

(FOC4255) DBCTL TERMINATION FAILED WITH

(FOC4256) DBCTL SESSION HAS NOT BEEN TERMINATED

(FOC4257) OPEN THREAD FAILED WITH

(FOC4258) TERMINATE THREAD FAILED WITH

(FOC4259) THE DATA IS INCOMPLETE: DBCTL SESSION HAS BEEN RECYCLED

(FOC4260) SET IS INVALID, FOCPSB IS NOT PARTITIONED

(FOC4261) FOCPSB MEMBER NOT FOUND

(FOC4262) FOCPSB IS NOT ALLOCATED

(FOC4263) FOCPSB IS ALLOCATED TO AN INVALID MEMBER OF A PDS

(FOC4264) INVALID SET OPTION. IMSMODE CANNOT BE SET VIA THIS COMMAND

(FOC4265) SECURITY CHECK FAILED. RESOURCE CLASS MISSING

(FOC4266) SECURITY CHECK FAILED. INSUFFICIENT AUTHORITY

(FOC4267) SET RESOURCE CLASS FAILED. INSUFFICIENT AUTHORITY
Security profile is owned by the server, individual users cannot change it.

(FOC4268) SET SECURITY FAILED. INSUFFICIENT AUTHORITY

(FOC4269) REQUEST TO CHANGE DFSPZP SUFFIX WAS DENIED
Individual users are not allowed to change PZP suffix in EDA/MSO environment.

(FOC4270) THIS REQUEST IS RUNNING AGAINST INCOMPATIBLE SERVER
Client is running OLD IMS and the XMI server is NEW, or client is running the
NEW IMS interface (IMSX) and the XMI server is OLD.

(FOC4271) FOCPSB HAS MORE PCB ENTRIES THAN THE ACTUAL PSB
The number of PCB entries in the FOCPSB file is greater than the number of
PCBs in the PSBGEN of the active PSB.

(FOC4272) CALLIMS ERROR ON CHNG CALL

(FOC4273) CALLIMS ERROR OCCURRED

(FOC4274) CALLIMS CRITICAL ERROR

(FOC4275) INVALID IMS MODE ON SERVER
The only valid modes on a server are REMOTE or DBCTL.

(FOC4276) DBCTL HAS BEEN INITIALIZED

Interface Errors and Messages

- (FOC4277) **KEY FIELD MISSING FROM SEGMENT:**
The segment is defined as keyed, but there is no .KEY field described in the MFD for that segment. Database will be processed sequentially.
- (FOC4278)
- (FOC4279)
- (FOC4280) **IMSMODE**
- (FOC4281) **PSB**
- (FOC4282) **IMSSEC**
- (FOC4283) **IMSCLASS**
- (FOC4284) **IMSPZP**
- (FOC4285) **DELAYED SHUTDOWN IS IN PROGRESS FOR DBCTL INTERFACE**
- (FOC4286) **DBCTL SHUTDOWN IS DELAYED, NUMBER OF ACTIVE REQUESTS**
- (FOC4287) **Reserved for IMD**
- (FOC4288) **DBCTL INTERFACE IS**
This message will be accompanied by additional DBCTL messages that describe the situation.
- (FOC4289) **DFSPZP SUFFIX**
- (FOC4290) **IMS SECURITY IS**
- (FOC4291) **IMS RESOURCE CLASS**
- (FOC4292) **NUMBER OF OPEN DBCTL THREADS**
- (FOC4293) **NUMBER OF ACTIVE DBCTL REQUESTS**
- (FOC4294) **Reserved for IMD**
- (FOC4295) **ACCESS POINTS TO DIFFERENT PSBS IN JOIN**
All files in a request should point to the same PSBname.
- (FOC4296)
- (FOC4297)
- (FOC4298)
- (FOC4299)

E.2 Common User Errors

This Section identifies common errors, their causes, and corrective measures.

1. (FOC263) EXTERNAL FUNCTION OR LOAD MODULE NOT FOUND: IMSX

Explanation:

The Interface load module, member IMSX in the 'prefix.IMS.LOAD' library, must be allocated to ddname STEPLIB or USERLIB. This error is usually caused by improper installation of the IMS/DB Interface. The installation instructions state that module IMSX must be allocated as ddname USERLIB or be concatenated into the allocation for ddname STEPLIB.

Solution:

Either allocate 'prefix.IMS.LOAD' as ddname USERLIB or concatenate it to ddname STEPLIB as described in Appendix D, *Installation Instructions*.

2. (FOC4213) LOCAL DLI CALL ERROR STATUS FOR SEGMENT xxxxxxxx/AC (IMS RETURN CODE AC: HIERARCHICAL ERROR IN SSA)

Explanation:

The Interface issued a call to IMS for the segment named xxxxxxxx, and IMS found an error in the hierarchical structure of the IMS database. There are three possibilities:

- The Master File does not *exactly* match the hierarchy described in the PCB. Use the PSBGEN source statements to check the PCB against the Master File.
- You are using an alternate view of the database and you have requested data from a segment other than the alternate view segment. FOCUS allows you to use TABLE FILE FILENAME.FIELDNAME for IMS files and generates the corresponding unqualified IMS calls for the appropriate segment. However, if you attempt to retrieve data from segments other than the one that is implied in the TABLE request, FOCUS issues the IMS calls using the alternate view structure. IMS will not interpret the call because you have dynamically altered the Master File and the description no longer *exactly* matches the PCB.
- You have not included all of the IMS segments from the PCB in the Master File. Each IMS segment to be retrieved must be defined in the Master File in the proper order.

Solution:

- Do not use alternate views when using an IMS database.
- Correct any errors in the Master File.

3. (FOC4213) LOCAL DLI CALL ERROR STATUS FOR SEGMENT xxxxxxxx/AD
(IMS RETURN CODE AD: INVALID PCB USED IN REQUEST EXECUTION)

Explanation:

The Interface used an invalid PCB in a call to IMS. This error occurs when the FOCPSB does not account for the CMPAT parameter in the PSB. Either of the following may be the cause:

- The request used a terminal PCB when it should have used a DB PCB.
- The request used a DB PCB when it should have used a terminal PCB.

Solution:

- If the PSB contains the parameter CMPAT=YES, insert an additional TERM PCB in the FOCPSB (see Chapter 3, *Creating FOCUS Descriptions*).
 - If the PSB does not contain the CMPAT=YES parameter, be sure there is no extra TERM PCB in the FOCPSB.
4. (FOC4213) LOCAL DLI CALL ERROR STATUS FOR SEGMENT xxxxxxxx/AI
(IMS RETURN CODE AI: DATABASE OPEN ERROR)

Explanation:

The Interface issued a call to IMS for the segment named xxxxxxxx and IMS was unable to open the database. There are three possibilities:

- The ddname does not match the name specified in the DBD.
- The database requested is an offline database and it was not allocated in your CLIST, batch JCL, or from within your session.
- You are attempting to access an online database in BMP mode and it was not allocated in the IMS message region.
- The DBD and PSB are out of synch.

Solution:

- a. Allocate the correct ddname in your JCL, CLIST, or from within your session.
- b. Check your execution mode. If you are running in DLI (batch) mode, allocate the database in your CLIST, batch JCL, or from within your session. If you are trying to report from an online database, see your System Support Group to allocate the database in the IMS region. Be certain that all datasets in the database are allocated. A good source for the allocations required for offline databases is the JCL used for another application program that accesses the database.
- c. Use the ACB instead of the PSB by executing DFSRRC00 with a PARM of DBB rather than DLI (see Chapter 5, *Environments*). Notify your IMS systems programmer.

5. (FOC4213) LOCAL DLI CALL ERROR STATUS FOR SEGMENT xxxxxxxx/AJ
(IMS RETURN CODE AJ: INVALID PARAMETER FORMAT IN I/O AREA)

Explanation:

Possible problem in Master File.

Solution:

- a. Check that the Master File does not define a GROUP within a GROUP.
 - b. If a GROUP is defined as a key or secondary index, check that the keyword ALIAS is specified explicitly in the Master File and that the ALIAS value includes the correct suffix if the GROUP field is a sequence field, search field, or secondary index (see Chapter 3, *Creating FOCUS Descriptions*).
 - c. Verify the segment layout and the sequence and search fields against the DBD.
6. (FOC4213) LOCAL DLI CALL ERROR STATUS FOR SEGMENT xxxxxxxx/AK
(IMS RETURN CODE AK: INVALID FIELD NAME IN CALL)

Explanation:

The Interface issued a call to IMS in order to retrieve the segment named xxxxxxxx, and IMS does not understand the segment names or the fieldnames in the call.

The names of the segments, fields, and keys in the Master File do not match the names in the IMS DBD. There are three possibilities:

- The segment names in the Master File do not exactly match the SEGM statements (segment names) in the IMS DBD.
- An ALIAS value in the Master File includes the .KEY or .HKY suffix, but either the fieldname does not exist in the DBD, the fieldname does exist in the DBD but it is not designated as the IMS key field (SEQ,U or SEQ,M), or the ALIAS value does not match the name in the DBD.
- An ALIAS in the Master File includes the .IMS suffix, but the exact field name does not appear in the DBD.

Note that IMS search fields must be defined in the DBD.

Solution:

- a. Remove the .IMS from any field that is not mentioned in the DBD.
- b. Remove the .KEY designation from those fields that are not key fields, or correct the spelling of the ALIAS value.
- c. Correct all segment names in the Master File to match the DBD SEGM names.

Interface Errors and Messages

7. USER ABEND 100 IN AN IMS XMI REGION

Explanation:

This abend results when ddname FOCBMP is not part of the IMS XMI JCL, or the dataset that is allocated to ddname FOCBMP is not catalogued.

Solution:

Correct the error and resubmit the job.

8. (FOC000) ERROR MESSAGE TEXT MISSING: nnnn

Explanation:

Error message number nnnn could not be found in the datasets allocated to ddname ERRORS. The Interface error messages may not have been copied to the error message dataset, as described in Appendix D, *Installation Instructions*.

Solution:

Either copy the error messages as described in Appendix D, *Installation Instructions*, or concatenate 'prefix.IMS.DATA' in your allocation for ddname ERRORS (prefix is the high-level qualifier for your FOCUS production libraries).

Glossary

Access File	An optional FOCUS file description available in the DBCTL environment. It identifies the PSB to use with a request.
ACB	Application Control Block. Optimized PSB. Contains consolidated information from the PSB and DBD. Speeds up online processing. Created by an ACBGEN.
BMP	Batch Message Processing. Batch job with access to online resources.
Concatenated key	Concatenated key values of all segments along the hierarchical path to a particular segment instance.
Control Region	IMS address space that controls the IMS system. Maintains order among terminals, databases, and application programs in the online environment.
CCTL	Coordinating Controller. Transaction manager used with a DBCTL.
CICS	Customer Information Control System.
Current database position	The segment retrieved by the most recent DL/I call.
Data sensitive	IMS has access to the data in the segment, not just to its key. See also <i>Key sensitive</i> .
DBCTL	DB Control. Control region used in a DB-only system.
DBD	Database Description. Physical structure of the database. Created by a DBDGEN.
DBRC	Database Recovery Control.
DDIR	DMB Directory. Directory of databases.
DEDB	Data Entry Database. A type of Fast Path database.
Dependent Region	IMS address space responsible for processing transactions scheduled for it by the control region. The application program is executed in this region.
DL/I	Data Language I (one). Language for accessing IMS databases.
DLISAS	DL/I Separate Address Space. Address space responsible for database handling.
DMB	Data Management Block. Optimized DBD.
EMH	Expedited Message Handling. Fast Path transactions.
Fast Path	Special IMS access methods that provide enhanced data reliability, availability, and improved response time, but place limitations on the database structure and ability to take advantage of techniques like secondary indexing and logical relationships.
Field level sensitivity	Only the specific fields listed in the PCB are accessible to the application program. If no fields are listed in the PCB, the entire segment is accessible.

FOCPSB	A required FOCUS file description that describes the PSB and associates Master Files with the PCBs in the PSB.
GN	Get Next. DL/I sequential read.
GU	Get Unique. Non-sequential DL/I retrieval call.
HDAM	Hierarchical Direct Access Method. Uses a randomizer or hashing routine to process the root key. Access to dependent segments is through pointers. Random access to roots is quick, but IMS cannot retrieve them sequentially in root key order.
HIDAM	Hierarchical Indexed Direct Access Method. Access to root segments is through an index. The index is a separate database from the data. Can process records in root key sequence, but the need to read the index database as well as the data database slows access. Access to dependent segments is the same as in HDAM databases.
Hierarchical sequence	The order in which IMS accesses segments in a database. The hierarchical sequence proceeds from the root, to the first child of the root segment, to the child's first child, down to the lowest level; then it proceeds up one level and follows the next segment down through its children, and so on.
Hierarchical path	Consists of a segment and all of its ancestors starting from the root.
HISAM	Hierarchical Indexed Sequential Access Method. In a HISAM database there are two datasets: a primary indexed dataset that contains the root and most of the dependent segments from each record, and an overflow sequential dataset that contains the rest of each record. Segments within a record are associated by physical adjacency.
HSAM	Hierarchical Sequential Access Method. Relates segments by physically placing them in hierarchical order. Needs static data. A GU call produces a sequential scan from the beginning of the database.
IFP	IMS Fast Path. Region for EMH transactions.
IMS	Information Management System.
IMS/DB	IMS Database Manager. Stand-alone product in V4R1.
IMS/DC	IMS Database Communications. Old name for terminal/transaction handler.
IMS/ESA	IMS/Enterprise Systems Architecture (V3,V4). Designed for MVS/ESA.
IMS/TM	IMS Transaction Manager. New name for terminal/transaction handler.
Key	A field in a segment that defines the segment's order within its twin chain.
Key feedback area	Area of a PCB for storing the concatenated key of a segment retrieved by a DL/I call.
Key sensitive	IMS has access to the key of the segment, not the data in the segment. Used when a segment has no data of interest, but a lower level segment does. Indicated by PROCOPT=K in the PCB. See also <i>Data sensitive</i> .

IRLM	Inter-Region Lock Manager or IMS Resource Lock Manager.
Master File	A required FOCUS file description that describes the segments and fields accessible through a particular PCB.
Optimization	The process in which the Interface passes the record selection criteria of a FOCUS report request to IMS for processing.
PCB	Program Communication Block. There are two types: database and TP. A database PCB identifies a particular database and contains status information about calls made to the database. It also defines the segments within the database that the application program can access and the types of processing allowed. A TP PCB (also called an I/O PCB) contains status information about calls from the terminal it points to.
PSB	Program Specification Block. Defines the IMS resources needed by an application program. Contains PCBs. Created by a PSBGEN.
Qualified SSA	An SSA that contains selection criteria on search and/or sequence fields. IMS retrieves only those segments that satisfy the criteria.
Root segment	Unique segment at the top of the hierarchical tree.
Search field	A field listed in the DBD. Does not have to be a sequence field. It can be used in qualified SSAs.
Segment Code	A field in the segment that identifies its type. The DBDGEN assigns the type sequentially, starting with 01 for the root, and incrementing by 1 for each SEGM macro in the DBD.
Sensitive field	A field listed in the PCB. Fields omitted from the list are not available to the application program. If the PCB lists no fields, the whole segment is accessible.
Sequence field	Key field. Identifies a segment.
Short path	A segment that has no instance of its child segment type, or a segment in the host file of a join with no matching segment in the cross-referenced file.
SSA	Segment Search Argument. Describes the segments IMS should retrieve.
Subtree	A subset of the database that cannot include a segment without its parent.
Symbolic pointer	Concatenated key values of all segments along the hierarchical path up to and including the segment pointed to.
Twins	Segment instances of the same type and with the same parent.
Unique key	A key that fully identifies a segment. No other segment can have the same key value.
Unique segment	A segment with no twins; has a one-to-one relationship with its parent.
Unqualified SSA	An SSA that names the segment type to retrieve but does not specify additional selection criteria. Any segment of the named type satisfies the SSA.

Index

Symbol

.HKY, 2-15, 3-15, 3-20, 3-23

.IMS, 2-15, 3-15, 3-20, 3-23

.KEY, 2-15, 3-15, 3-20, 3-23

.SKY, 2-22, 3-22

? n command, E-1

? query command, 5-8, C-4

? RELEASE query, D-2

A

AC return code, E-7

ACB, 2-8, 5-13, 5-28

ACCEPT attribute, 3-27

Example, 3-27

ACCESS

Ddname, D-6

Library, D-6

Access File, 3-1, 3-34

Attributes

PSB, 3-34

Dataset, 3-34

Example

DI21PART, 3-34, A-3

Member of ACCESS library, D-6

Access method, 2-4

ACTUAL

Attribute, 3-18

Conversion chart, 3-18

Join format, 4-23, 4-25

AD return code, E-8

AGN, 6-1

AI return code, E-8

AIHDAM database, A-16

AJ return code, E-9

AK return code, E-9

ALIAS attribute, 2-15, 3-17

And SSA generation, 3-17

Name maximum, 3-17

Suffix value, 2-15, 3-15, 3-20, 3-23

For secondary index, 2-22, 3-22

ALL keyword, 4-23, 4-25

Allocation

Communication file, 5-21, D-19

MSO, D-20

FSTRACE, B-3

FSTRACE2, B-3

FSTRACE4, B-3

IMS.DATA, D-3, D-8

IMS.LOAD, D-3, D-7

Libraries, D-3

APPLCTN macro, D-10

Application Group Name, 6-1

Auto Index Selection, 4-18

And Join, 4-27

Order of precedence, 4-19

B

Batch. *See also* JCL

FSTRACE allocation, B-4

FSTRACE2 allocation, B-4

FSTRACE4 allocation, B-4

Unloading tape, D-4

BMP extension, C-10

Initiating, C-12

Invoking FOCUS, C-13

Terminating, C-14

- BMP mode
 - FOCUS loaded by DFSRRC00, 5-25
 - CLIST, 5-26
 - Diagram, 5-25
 - JCL, 5-27
 - XMI server, 5-11
 - Diagram, 5-11
 - JCL for initiating, 5-12

BSIZ parameter, 5-2

C

Chart

- ACTUAL format, 3-18
- Environments, 5-31

CHECK FILE, 4-29

Checkpoint, 3-4

CICS

- DFHLIB, D-21
- XMI server
 - Diagram, 5-17
 - JCL for initiating, 5-18
 - Restriction, 5-17

CLIST for invoking FOCUS

- DBCTL, 5-6
- FOCUS loaded by region controller
 - BMP mode, 5-26
 - DLI mode, 5-29
 - XMI server, 5-19

CMPAT, 3-4

CMS, 5-3

- Accessing the Interface, C-15
- And IMS databases, C-15

CNBA parameter, 5-3

COBOL FD

- Using to create Master File, 3-1

Command

- Console, D-16
- Interface environmental, C-1

Command code, 2-9

Communication file, 5-10, 5-20, C-11

- Concurrent XMI jobs, 5-21
- Installation, D-19

Concatenated key, 2-3

Concatenated PCB, 3-7

- Example, 3-7

CONCATNAME attribute, 3-7

Configuration file

- Attributes, D-14
- Security, 6-2
- Sample, D-14

Console command, D-16

D

Data sensitive, 3-13

Database position, 2-6

Database sharing, 5-17

DBA security, 6-1

DBB parameter, 5-13, 5-28

DBBF parameter, 5-2

DBCTL, 5-3, 5-8, C-3

- Advantages, 5-4
- Attribute, D-14
- CLIST, 5-6
- Diagram, 5-5
- FSTRACE4 example, B-13
- Installation, D-8

- APPLCTN macro, D-10

- Console commands, D-16

- DRA Startup Table, D-10

- IMDTST, D-9

- Keywords, D-12

- MSO server JCL, D-15

- Sample configuration file, D-14

- Testing installation, D-9

DBCTL (*continued*)
 Invoking, 5-6, 5-7
 JCL, 5-7
 Security, 6-1
 Configuration file, 6-2
 RACF, 6-2
 SAF interface, 6-2
 Shutting down, D-17
 With MSO, C-3

DBD, 2-5
 DBDGEN, 2-5
 Example, A-1
 AIHDAM, A-16
 EMPDB, A-8
 PATDB01, A-3
 FIELD, 2-5
 SEGM, 2-5
 XDFLD, 2-22, 3-22, C-5

DBDGEN, 2-5

DBDNAME, 2-6

DBFX parameter, 5-2

DBRC, 5-17

DCB parameters, D-4, D-5

Debugging techniques, B-1

DEDB, 2-4
 Optimization, 4-6

DFHDRP, 5-24
 Restriction, 5-17

DFHLIB, D-21

DFSPZPxx, D-10

DFSRRC00, 5-11, 5-14, 5-25, 5-28

DI21PART database, A-1

Diagram
 DBCTL, 5-5
 FOCUS loaded by DFSRRC00
 BMP mode, 5-25
 DLI mode, 5-28
 XMI server
 BMP mode, 5-11
 CICS, 5-17
 DLI mode, 5-13

Distribution tape, D-3
 Identifying maintenance level, D-2
 Identifying release level, D-2
 Unloading with JCL, D-4

DL/I call, 2-8, 4-3
 Generated by Interface, 1-2
 GN, 2-9
 GU, 2-9
 Knowledge of, D-1
 Path call, 4-3
 PCB requirements, 4-3
 Subtree retrieval, 4-3

DLI mode
 BMP extension
 JCL, C-12
 FOCUS loaded by DFSRRC00, 5-28
 CLIST for invoking, 5-29
 Diagram, 5-28
 JCL for invoking, 5-30
 XMI server, 5-13
 Diagram, 5-13
 JCL for initiating, 5-14

DLITRACE, B-2

DRA Startup Table, D-10
 Keywords, D-12

Dump format, 4-5, B-12

Duplicate PCBs, 3-5, 5-22

Dynamic join, 2-24. *See also* Join

Dynamic PCB selection exit, 3-9

E

- Efficiency, 1-5
- Embedded join, 2-24
- EMPDB databases, A-8
- Environment, 5-1
 - Acceptable, 1-3
 - BMP extension, C-10
 - DBCTL, 5-3
 - Fast Path considerations, 5-2
 - Fast Path parameters, 5-12, 5-26, 5-27
 - FOCUS loaded by DFHDRP, 5-24
 - FOCUS loaded by DFSRRC00, 5-24
 - BMP mode, 5-25
 - DLI mode, 5-28
 - PSB
 - PROCOPT, 5-3
 - Summary chart, 5-31
 - Switching, 5-33
 - To DBCTL, 5-33
 - To XMI server, 5-34
 - XMI server, 5-9
 - BMP mode, 5-11
 - CICS, 5-16
 - CICS restriction, 5-17
 - Communication file, 5-20
 - DLI mode, 5-13
 - JCL for initiating, 5-12, 5-14, 5-18
- Environmental command, C-1
- Error handling, B-1
 - DLITRACE, B-2
 - FSTRACE, B-3
 - FSTRACE2, B-3
 - FSTRACE4, B-3
 - Return code, B-3
- Error message, E-1
 - ? n command, E-1
 - Common user errors, E-7
 - Return code, E-7
 - AC, E-7
 - AD, E-8
 - AI, E-8
 - AJ, E-9
 - AK, E-9
- ERRORS library, D-8
- Example
 - ACCEPT attribute, 3-27
 - Access File, 3-34
 - DI21PART, A-3
 - Concatenated PCB, 3-7
 - Configuration file, D-14
 - DBD
 - AIHDAM, A-16
 - EMPDB, A-8
 - PATDB01, A-3
 - File Description, A-1
 - AIHDAM, A-16
 - DI21PART, A-1
 - EMPDB, A-8
 - PATDB01, A-3
 - FOCPSB
 - AIHDAM, A-17
 - DI21PART, A-2
 - EMPDB, A-11
 - PATDB01, A-6
 - FSTRACE, B-5
 - FSTRACE4, 4-9, 4-11, B-13
 - Dump format, 4-4
 - Group field, 3-21
 - Join, 4-30
 - Optimization, 4-27

Example (*continued*)

Master File

AIHDAM, A-17

DI21PART, A-2

EMPDB, A-12

PATDB01, A-7

MISO JCL, D-15

OCCURS=fieldname, 3-32

OCCURS=n, 3-31

OCCURS=VARIABLE, 3-33

Optimization, 4-4

ORDER field, 3-30

Partitioned PCB, 3-7

PSB

AIHDAM, A-17

DI21PART, A-2

EMPDB, A-11

PATDB01, A-6

Record selection test

Multiple segments, 4-16

Multiple SSAs, 4-11

Partial key, 4-15

READLIMIT, 4-21

Secondary index, 4-18

Single SSA, 4-9

WHERE test, 4-13, 4-14

RECTYPE, 3-26

F

Fast Path considerations, 5-2

BSIZ, 5-2

CNBA, 5-3

DBBF, 5-2

DBFX, 5-2

NBA, 5-2, 5-12, 5-26, 5-27

OBA, 5-2, 5-12, 5-26, 5-27

Field

Attributes, 3-14, 3-21

Length limit, 3-19

Name maximum, 3-17

Naming conventions in Master File, 3-16

Qualifier, 3-16

FIELD in DBD, 2-5

Field level sensitivity, 2-7

FIELDNAME, 3-16

Attribute, 2-15, 3-15, 3-16

Maximum length, 3-17

FILE attribute, 3-12

File attributes, 3-12

File Description samples, A-1

AIHDAM, A-16

DI21PART, A-1

EMPDB, A-8

PATDB01, A-3

FILENAME attribute, 2-12

FOCBMP file, 5-10, 5-20, C-11

Allocation, 5-21

Concurrent XMI jobs, 5-21

FOCBMPKX, C-14

FOCEXEC

Library, D-7

FOCPSB, 3-2

And secondary index, 3-24

Attributes

CONCATNAME, 3-7

FOCPSB, 3-3

LOWVALUE, 3-6

PCBNAME, 2-10, 3-4

PCBTYPE, 2-10, 3-4

PL1, 3-3

USE, 3-7

Concatenated PCB, 3-7

Dataset, 3-8

Ddname, D-6

Example, A-1

AIHDAM, A-17

DI21PART, A-2

EMPDB, A-11

PATDB01, A-6

Secondary index, A-6, A-17

FOCPSB (*continued*)

- Fixed format, C-10
- Header record, 3-3
- Library, D-6
- Mapping concepts, 2-10
 - Secondary index, 2-21
- Member of FOCPSB library, D-6
- Partitioned PCB, 3-5
 - Request processing, 3-8

FOCUS

- Field qualifiers, 3-16
- Installation requirement, D-2
- Invoking
 - BMP extension, C-13
 - FOCUS loaded by DFSRRC00, 5-26, 5-29
 - XMI server, 5-19, 5-20
- Knowledge of, D-1
- PTF, D-2

FOCUS loaded by DFHDRP, 5-24

FOCUS loaded by DFSRRC00, 5-24

- BMP mode, 5-25
 - CLIST, 5-26
 - Diagram, 5-25
 - JCL, 5-27
- DLI mode, 5-28
 - CLIST, 5-29
 - Diagram, 5-28
 - JCL, 5-30

Format in a join, 4-23, 4-25

FSTRACE, B-3

- Allocating, B-3
- Batch mode, B-4
- Disabling, B-4
- Example, B-5

FSTRACE2, B-3

- Allocating, B-3
- Batch mode, B-4
- Disabling, B-4

FSTRACE4, B-3

- Allocating, B-3
- Batch mode, B-4
- Disabling, B-4
- Example, 4-9, 4-11, B-13
 - Dump format, 4-4

G

Get

- Next, 2-9
- Unique, 2-9

GN DL/I call, 2-9

GO PROCOPT, 5-3

GROUP, 2-17, 2-22, 3-21, 3-22

- And packed fields, 3-20
- And SSA generation, 3-21
- Example, 3-21

GU DL/I call, 2-9

H

HDAM, 2-4

- Optimization, 4-6
- Partitioned PCB, 3-6

HIDAM, 2-4

- Optimization, 4-6

Hierarchical

- Path, 2-2
- Sequence, 2-2
- Structure, 2-2

HISAM, 2-4

How the Interface works, 1-2

HSAM, 2-4

- I
-
- I/O PCB, 3-4
 - CMPAT, 3-4
 - IF test, 4-6. *See also* Record selection test
 - IMDTST, B-11, D-9
 - IMDYNPCB exit, 3-9
 - IMS, C-1
 - Access methods, 2-4
 - Accessing from CMS, C-15
 - Concepts, 2-1
 - Concatenated key, 2-3
 - Field level sensitivity, 2-7
 - Hierarchical path, 2-2
 - Hierarchical sequence, 2-2
 - Hierarchical structure, 2-2
 - Join, 2-24
 - Key feedback area, 2-8
 - Key field, 2-2
 - Key sensitivity, 2-7
 - Logical relation, 2-3
 - Search field, 2-3
 - Secondary index, 2-3
 - Sequence field, 2-2
 - SSA, 2-9
 - Status code, 2-8
 - Symbolic pointer, 2-3
 - Twin, 2-2
 - Control block, 2-5
 - ACB, 2-8
 - DBD, 2-5
 - PCB, 2-6
 - PSB, 2-6
 - Default value, C-1
 - DL/I call, 2-8
 - Generated by Interface, 1-2
 - GN, 2-9
 - GU, 2-9
 - Installation requirement, D-2
 - Join, 2-24
 - Overview, 2-1
 - IMS (continued)
 - Return code, E-7
 - AC, E-7
 - AD, E-8
 - AI, E-8
 - AJ, E-9
 - AK, E-9
 - Trace, B-2
 - IMS.DATA, D-4, D-8
 - File on distribution tape, D-3
 - IMS.LOAD, D-4, D-7
 - File on distribution tape, D-3
 - IMSCCLASS attribute, 6-2, D-14
 - IMSECHK installation, 6-6
 - IMSECHK security exit, 6-3
 - IMSECHK tracing, 6-6
 - IMSNONSTOP, 5-35
 - IMSPZP, 5-8, C-3
 - Attribute, D-14
 - IMSSEC attribute, 6-2, D-14
 - Installation, D-1
 - DBCTL, D-8
 - APPLCTN macro, D-10
 - Console commands, D-16
 - DRA Startup Table, D-10
 - IMDTST, D-9
 - Keywords, D-12
 - Sample configuration file, D-14
 - Sample JCL, D-15
 - Testing installation, D-9
 - Libraries
 - ACCESS, D-6
 - Allocating, D-3
 - ERRORS, D-8
 - FOCEXEC, D-7
 - FOCPSB, D-6

Installation

Libraries (*continued*)

Interface load library, D-7

Interface maintenance, D-8

MASTER, D-5

Preparing, D-5

Maintenance, D-2

Level, D-2

Release level, D-2

Overview, D-3

Prerequisite, D-1

Disk space, D-4

Software, D-2

Space allocation, D-4

Run-time requirement, D-22

Unloading distribution tape, D-4

XMI server, D-18

Communication files, D-19

JCL, D-20

J

JCL, D-3

BMP extension, C-12

FSTRACE allocation, B-4

FSTRACE2 allocation, B-4

FSTRACE4 allocation, B-4

Initiating XMI server

BMP mode, 5-12

CICS, 5-18

DLI mode, 5-14

Invoking FOCUS

BMP extension, C-13

DBCTL, 5-7

FOCUS loaded by DFSRRC00, 5-27, 5-30

XMI server, 5-20

Job card, D-4

MSO sample, D-15

Unloading distribution tape, D-4

XMI server, D-20

Join, 4-22

ACTUAL format, 4-23, 4-25

And Auto Index Selection, 4-27

CLEAR, 4-23, 4-26

Dynamic, 2-24

Embedded, 2-24

Example, 4-24, 4-30

IMS, 2-24

Multi-field, 4-25

Multiple, 4-23, 4-25

Non-unique, 4-23, 4-25

Optimization, 4-27

And SQL Translator, 4-27

Partial key, 4-26

PSB restriction, 4-24, 4-26

Recursive, 4-24, 4-26

Single-field, 4-23

Types of, 4-22

Unique, 4-23, 4-25

With selection criteria, 4-31

K

Key feedback area, 2-8

Key field, 2-2

Key sensitive, 2-7, 3-13

Keyword

Configuration file, D-14

DRA Startup Table, D-12

L

Library

ACCESS, D-6

Allocating, D-3

DCB parameters, D-4, D-5

ERRORS, D-8

FOCEXEC, D-7

FOCPSB, D-6

IMS.DATA, D-3, D-8

IMS.DATA(FOCIMS), D-8

IMS.LOAD, D-3, D-7

Interface load library, D-7

Library (*continued*)

- Interface maintenance, D-8
- MASTER, D-5
- Private and public, D-5
- Run-time, D-5

Limit

- Field length, 3-11, 3-19
- Levels, 3-11
- Segment, 3-11

Limitation

- Field naming conventions, 3-16
- Fieldname maximum, 3-17
- File naming conventions, 3-12
- Qualified fieldnames and OCCURS, 3-17
- Write operation, 1-1

LIST, PCB, 3-4

Logical relation, 2-3

Logical segment types

- RECTYPES, 2-19

LOWVALUE attribute, 3-6

M

Maintenance

- Level, D-2
- PTF, D-2
- Reinstallation, D-2
- Release level, D-2

Mapping concepts, 2-10

- Dynamic join, 2-24
- FOCPSB, 2-10
 - Secondary index, 2-21

Master File, 2-11

- ALIAS, 2-15
- FIELDNAME, 2-15
- FILENAME, 2-12
- GROUP, 2-17
- OCCURS, 2-20
- PARENT, 2-13
- RECTYPE, 2-17
- Secondary index, 2-22, C-5

Mapping concepts

Master File (*continued*)

- Segment record, 2-14
- SEGNAME, 2-13
- SEGTYPE, 2-13
- Subtree requirement, 2-12

Master File, 3-11

- ALIAS, 3-17
 - Suffix value, 2-15, 2-22, 3-22

Attributes

- ACCEPT, 3-27
- ACTUAL, 3-18
- ALIAS, 2-15, 3-15, 3-17, 3-20, 3-23
- FIELDNAME, 2-15, 3-15, 3-16
- FILE, 3-12
- FILENAME, 2-12, 3-12
- GROUP, 3-21
- MISSING, 3-15
- OCCURS, 2-20, 3-28
- ORDER, 3-30
- PARENT, 2-13, 3-14
- POSITION, 3-28
- RECTYPE, 2-17, 3-25
- SEGNAME, 2-13, 3-12, 3-13
- SEGTYPE, 2-13, 3-12, 3-13
- SUFFIX, 3-12
- USAGE, 3-15, 3-18, 3-27

Creating with COBOL FD Translator, 3-1

Dataset, 3-11

Example, A-1

- ACCEPT attribute, 3-27
- AIHDAM, A-17
- Concatenated PCB, 3-7
- DI21PART, A-2
- EMPDB, A-12
- Group field, 3-21
- OCCURS=fieldname, 3-32
- OCCURS=n, 3-31
- OCCURS=VARIABLE, 3-33
- ORDER field, 3-30, 3-33
- Partitioned PCB, 3-7
- PATDB01, A-7
- RECTYPE, 3-26
- Secondary index, A-7

Master File (*continued*)

- Field attributes, 3-14
- Field length limit, 3-11
- Field naming conventions, 3-16
- File attributes, 3-12
- Limits, 3-11
- Mapping concepts, 2-11
 - ALIAS, 2-15
 - FIELDNAME, 2-15
 - FILENAME, 2-12
 - GROUP, 2-17
 - OCCURS, 2-20
 - PARENT, 2-13
 - RECTYPE, 2-17
 - Secondary index, 2-22, C-5
 - Segment record, 2-14
 - SEGNAME, 2-13
 - SEGTYPE, 2-13
 - Subtree requirement, 2-12
 - Variable length segment, 2-20
- Member of MASTER library, D-5
- Naming conventions, 3-12
- Rules for declarations, 3-11
- Secondary index, 3-22
- Segment attributes, 3-12
- Segment limit, 3-11
- Subtree requirement, 3-1
- Suffix value, 3-15, 3-20, 3-22, 3-23

MASTER library, D-5

Message. *See* Error message

MISSING

- Attribute, 3-15
- Test, 4-4

MSDB, 2-5

- MSO, 5-3, 5-10, 5-11, 5-17, D-1
 - Allocating communication file, D-20
 - Configuration file sample, D-14
 - Console commands, D-16
 - DBCTL attribute, D-14
 - IMSCLASS attribute, D-14
 - IMSPZP attribute, D-14

MSO (*continued*)

- IMSSEC attribute, D-14
- JCL sample, D-15
- Security, 6-2
- With DBCTL, C-3

Multi-field join, 4-25

Multiple

- Segments
 - Record selection example, 4-16
- SSAs
 - Constructing, 4-10
 - Record selection example, 4-11
- XMI server jobs, 5-21

N

Naming conventions for fields, 3-16

NBA parameter, 5-2, 5-12, 5-26, 5-27

New Interface

- Activating, C-1

NONSTOP, 5-35

Non-stop processing, 5-34

O

OBA parameter, 5-2, 5-12, 5-26, 5-27

OCCURS segment, 2-20, 3-28

- Example, 3-31
- Field qualifier limitation, 3-17
- ORDER, 3-30
- POSITION, 3-28
- VARIABLE, 3-28

Optimization, 1-2, 1-5, 4-2

- DEDB, 4-6
- Example, 4-9
- HDAM, 4-6
- HIDAM, 4-6
- IF test, 4-6
 - Rules, 4-6
- Join, 4-27
- Multiple SSAs, 4-10

Optimization (*continued*)

- Relations, 4-7
- Rules, 4-2
- Single SSA, 4-9
- SSA buffer, 4-8
- WHERE test, 4-13

ORDER field, 3-30

- Example, 3-30

Overview

- IMS concepts, 2-1

P

Packed field

- In GROUP, 3-20

PARENT, 2-6, 2-13, 3-14

Partial key

- Join, 4-26
- Record selection example, 4-15

Partitioned PCB, 3-5

- Example, 3-7
- HDAM, 3-6
- Request processing, 3-8

PATDB01 database, A-3

Path call, 4-3

PCB, 2-6

- Concatenated, 3-7
 - Example, 3-7
- Database position, 2-6
- DBDNAME, 2-6
- Duplicates in PSB, 3-5, 5-22
- Example, A-1
- I/O, 3-4
 - CMPAT, 3-4
- LIST, 3-4
- PARENT, 2-6
- Partitioned, 3-5
 - Example, 3-7
 - Request processing, 3-8
- PROCOPT, 2-6, 3-3, 5-3

PCB (*continued*)

- PROCSEQ, 2-6, C-5, C-7
- SENFLD, 2-14, 3-14
- SENSEG, 2-6
- Status code, 2-6
- XMI server, 5-22

PCBNAME attribute, 2-10, 3-4

PCBTYPE attribute, 2-10, 3-4

- TERM, 3-4

PCICSPSB, 6-2

PL1 attribute, 3-3

POSITION attribute, 3-28

prefix, D-4

Processing overview, 1-2

PROCOPT, 2-6, 3-3

- For Interface, 5-3

PROCSEQ, 2-6, 3-22, C-5

PSB, 2-6, 5-8, C-3

- And Join, 4-24, 4-26
- Attribute, 3-34
- Duplicate PCBs, 3-5
- Example, A-1
 - AIHDAM, A-17
 - DI21PART, A-2
 - EMPDB, A-11
 - PATDB01, A-6
 - Secondary index, A-6
- PROCOPT for Interface, 5-3
- PSBGEN, 2-6
- Selecting
 - Access File, 3-1
 - In DBCTL, 5-8
 - XMI server, 5-22

PSBGEN, 2-6

PTF, D-2

PUT level. *See* Maintenance level

Q

- Qualified SSA, 2-9, 4-2
 - On lower level segment, 4-18
- Qualifier, 3-16
- Query command, 5-8, C-4

R

- RACF, 6-2
 - PCICSPSB, 6-2
- Record selection test
 - And Join, 4-27
 - IF, 4-6
 - MISSING, 4-4
 - Multiple SSAs, 4-10
 - Optimized relations, 4-7
 - Secondary index, 4-18
 - Single SSA, 4-9
 - SSA buffer, 4-8
 - WHERE, 4-13
- RECTYPE, 2-19
- RECTYPE attribute, 2-17, 3-25
 - Example, 3-26
- Recursive join, 4-24, 4-26
- Release level, D-2
- Report
 - Facilities available, 1-4
- Report Writer, 1-2
- Reporting efficiencies, 4-1
- Request
 - Processing, 3-2
- Retrieval
 - Example with FSTRACE, B-5
 - Example with FSTRACE4, B-13
 - Sequential, 4-5
 - Unique segment, 3-13
 - CHECK FILE, 4-29

- Return code, E-7
 - AC, E-7
 - AD, E-8
 - AI, E-8
 - AJ, E-9
 - AK, E-9
 - Displaying with FSTRACE, B-3
 - FSTRACE example, B-11

- Root segment
 - Sequential access, 4-13

- Rule
 - Master File declarations, 3-11
 - Optimization, 4-6

- Run-time requirement, D-22
 - Library, D-5

S

- SAF interface, 6-2
- Sample
 - File Description, A-1
- Screening condition. *See* Record selection test
- Search field, 2-3
- Secondary index, 2-3, 2-21
 - And Join, 4-27
 - Auto Index Selection
 - Order of precedence, 4-19
 - Example, 4-18
 - In FOCPSB, 2-21, 3-24
 - In Master File, 2-22, 3-22, C-5
 - Record selection example, 4-18
 - Suffix value, 3-22
- Security, 6-1
 - DBA, 6-1
 - DBCTL
 - Configuration file, 6-2
 - RACF, 6-2
 - SAF interface, 6-2
 - Overview, 1-5

- Security exit
 - IMSECHK, 6-3
- SEGM in DBD, 2-5
- Segment
 - Attributes
 - Master File, 3-12
 - Record in MFD, 2-14
 - Unique, 3-12
 - Retrieval, 3-13, 4-29
- SEGNAME attribute, 2-13, 3-12, 3-13
- SEGTYPE attribute, 2-13, 3-12, 3-13
- Selection test. *See* Record selection test
- SENFLD, 2-14, 3-14
- SENSEG, 2-6
- Sensitivity
 - Data, 3-13
 - Field, 2-7
 - Key, 2-7, 3-13
- Sequence field, 2-2
- Sequential retrieval, 4-5
- Sequentially accessed root segment, 4-13
- SET
 - ?, 5-8, C-4
 - DBCTL, 5-8, C-3
 - FIELDNAME, 3-16
 - IMS, C-1
 - Default value, C-1
 - IMSNONSTOP, 5-35
 - IMSPZP, 5-8, C-3
 - NONSTOP, 5-35
 - PSB, 5-8, C-3
 - SSAOPT, 4-31
- Shared database environment, 4-1, 5-17
- Short path, 4-29
- Single SSA
 - Constructing, 4-9
 - Record selection example, 4-9

- Single-field join, 4-23
- SQL Translator
 - And join optimization, 4-27
- SSA, 2-9
 - Buffer, 4-8
 - Command code, 2-9
 - Dump format, 4-5, B-12
 - Example, 4-2, 4-5, 4-9, 4-11
 - Generation
 - And ALIAS, 3-17
 - And GROUP fields, 3-21
 - On lower level segment, 4-18
 - Qualified, 4-2
- SSAOPT, 4-31
- Status code, 2-6, 2-8. *See also* Return code
 - FSTRACE example, B-12
- Subtree
 - Requirement, 2-12
 - Retrieval, 4-3
- SUFFIX attribute, 3-12
- Suffix value, 2-15, 3-15, 3-17, 3-20, 3-23
- Switching access mode, 5-34
- Symbolic pointer, 2-3

T

- TERM, 3-4
- Termination
 - BMP extension, C-14
 - XMI server, 5-23
- Traces, B-1
 - Disabling, B-4
 - DLITRACE, B-2
 - FSTRACE, B-3
 - Allocating, B-3
 - Batch mode, B-4
 - Disabling, B-4
 - Example, B-5

Traces (*continued*)

- FSTRACE2, B-3
 - Allocating, B-3
 - Batch mode, B-4
 - Disabling, B-4
- FSTRACE4, B-3
 - Allocating, B-3
 - Batch mode, B-4
 - Disabling, B-4
 - Example, 4-5, 4-9, 4-11, B-13

Twin, 2-2

U

- Unique segment, 3-12, 3-13
 - Retrieval, 3-13
 - CHECK FILE, 4-29
- UNIT value, D-4
- Unqualified SSA, 2-9, 4-14, 4-17
- USAGE attribute, 3-15, 3-18, 3-27
- USE attribute, 3-7

V

- Variable length segment, 2-20
- VARIABLE, OCCURS value, 3-28
- VOL=SER value, D-4

W

- WHERE test, 4-13. *See also* Record selection test
- Write operation, 1-1

X

- XDFLD, 2-22, 3-22, C-5
- XMI server, 5-9
 - BMP mode, 5-11
 - Diagram, 5-11
 - JCL for initiating, 5-12
 - CICS
 - Diagram, 5-17
 - JCL for initiating, 5-18
 - Restriction, 5-17
 - Communication file, 5-20
 - Concurrent XMI jobs, 5-21
 - DLI mode, 5-13
 - Diagram, 5-13
 - JCL for initiating, 5-14
 - Installation, D-18
 - Communication file, D-19
 - JCL, D-20
 - Invoking FOCUS
 - CLIST, 5-19
 - JCL, 5-20
 - MSO allocation, D-20
 - Multiple jobs, 5-21
 - PCB, 5-22
 - PSB, 5-22
 - Shutting down, D-21
 - Terminating, 5-23
- XMIKILL, 5-23

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. Send comments to:

Corporate Publications
Attn: Manager of Documentation Services
Information Builders
1250 Broadway
New York, NY 10001-3782

or FAX this page to (212) 967-6406 or call Josephine Moscato at (212) 736-4433, x3670.

Name: _____

Company: _____

Address: _____

Telephone: _____ Date: _____

Comments:

Reader Comments
