OMNIDEX

Programmer's Guide

Version 2.05/2.06

October 1990

DISC

Dynamic Information Systems Corporation
Corporate Phone: (303) 893-0335
Technical Support Phone: (303) 592-1220
Technical Support FAX: (303) 892-5713
Sales Support FAX: (303) 893-2415

This manual was written with Microsoft Word 5.0 and formatted with Ventura Publisher Professional Extension 2.0 (Xerox). Screen captures were made using Hotshot (Symsoft), while in Reflection (Walker, Richer and Quinn).

Acknowledgements

Dynamic Information Systems Corporation would like to acknowledge the following products which are mentioned in this manual, and their distributors who are listed below alphabetically:

- ❑ Adager is a registered trademark of Adager.
- ❑ MPEX/3000 is a registered trademark of VESOFT Inc.
- ❑ The following are registered trademarks of Cognos Corporation: Powerhouse, QDD, QTP, Quiz and Quick.
- ❑ The following are registered trademarks of Hewlett-Packard Co.: BASIC/3000, Cobol II/3000, Compatibility Mode, DBChange, Dictionary 3000, Editor, FORTRAN/3000, HP, HP3000, HP FORTRAN/77, HPPA, HP Word, IMAGE, MPE V, MPE XL, Native Mode, PASCAL/3000, Query, Rapid, RPG, SORT, Spectrum, SPL, Transact and TurboIMAGE.
- ❑ The following are registered trademarks of Robelle Consulting Ltd.: Suprtool and Qedit.
- ❑ SortMate is a registered trademark of Running Mate.
- ❑ Hotshot is a registered trademark of Symsoft.
- ❑ Reflection is a registered trademark of Walker Richer and Quinn.
- ❑ The IMAGE/3000 Handbook is copyrighted by Wordware.
- ❑ ASK and MANMAN are registered trademarks of Ask Computer Systems Inc.

This page left blank intentionally

Table of Contents

Introduction

Programming

Appendix

Table of Illustrations

Chapter Overview

Welcome to the **OMNIDEX Information Management System**, the powerful software tool that adds state-of-the-art indexing capabilities plus extremely fast and flexible information retrievals to IMAGE, TurboIMAGE and TurboIMAGE/XL data bases.

OMNIDEX gives your users instant access to the information they need, through a natural, easy-to-use interface. Because it supports Boolean and relational operations through its retrieval intrinsics, it enables you to build better applications.

This chapter provides an overview of this manual and how to use it to implement the OMNIDEX Information Management System. It also provides a general overview of the OMNIDEX software and its capabilities, and information about using the on-line documentation system.

Note that all references to IMAGE in this manual apply to IMAGE, TurboIMAGE and TurboIMAGE/XL unless otherwise noted.

Using this Manual

The prerequisite for using this manual is that you have experience with IMAGE/3000, TurboIMAGE or TurboIMAGE/XL.

This manual consists of three chapters and an **Appendix**.

- ❑ This **Introduction** chapter provides an overview on the OMNIDEX Information Management System and how to use the documentation.
- ❑ The **Intrinsics** chapter discusses the IMSAM and Omnidex intrinsics, and how to use them effectively in your application programs.
- ❑ The **Programming** chapter provides information about how to trap your existing programs' intrinsic calls to automatically use OMNIDEX update intrinsics. It also discusses how to call the Omnidex and IMSAM intrinsics from COBOL and FORTRAN.

  Also see the **Intrinsics** chapter for detailed information about each intrinsic, and look at the sample programs provided in the DISC account. The sample programs are listed in a file called **FILES.DOC.DISC**.
- ❑ The **Appendix** contains a section on **Troubleshooting**, as well as a glossary.

When you have read this manual, you will be able to implement OMNIDEX's fast and flexible information management capabilities through your application programs. A copy of this manual, and the OMNIDEX **Administrator's Guide**, has been provided on-line and can be accessed by entering the UDC ON-LINE at any MPE system (:) prompt. See the **On-Line Documentation System** section near the end of this chapter. This provides instant access to any information you may need to reference .

When evaluating OMNIDEX for the first time, it is also recommended that you go through the **Demonstration Guide** and the **OMNIDEX Primer**, if possible. Both offer automated demonstrations and exercises to familiarize you with the installation and retrieval capabilities of OMNIDEX. The **DATADEX Reference Guide** is useful reading after you've installed OMNIDEX on a data base. DATADEX can be used to query and test your prototype data base.

Conventions

Upper Case

UPPER CASE LETTERS are used for the names of intrinsics and commands. For example, DBIFIND is an intrinsic; INDEX is a command.

Also, acronyms, like IMSAM, for **IMAGE** Sequential Access Method; DBINSTAL, for Data **Base Installation**; and DBIUTIL for Data **Base** IMSAM Utility, appear in upper case letters.

References to chapter sections and subsections appear in upper and lower case letters. For example, this section is called **Conventions.**

Bold Print

Bold print is used for chapter titles, headings and items in a list.

Italics

New or special terms appear in italics when first used. For example, *generic retrieval* is mentioned in the **Intrinsics** chapter. Terms which appear in italics will be listed in the glossary for future reference.

Italics are also used to denote a variable or parameter that is specified differently depending on the individual application, as in:

```
filename
```

Here, *filename* represents an MPE file of indeterminate name.

Courier Font

Courier font is used for prompts and messages that appear on the terminal screen, information to be entered at prompts, or job and program files. For example:

```
This text is in Courier.
```

Usage

OMNIDEX in all capital letters refers to the entire OMNIDEX Information Management System (IMS). This manual refers primarily to OMNIDEX Version 2.06, though most of the information discussed also applies to OMNIDEX Version 2.05.

Version 2.05

Any information that applies only to Version 2.05 is highlighted by the icon that appears to the left of this paragraph, or discussed in a separate section. For example, **OMNIDEX Version 2.05 SL and XL Placement** is a section devoted exclusively to OMNIDEX Version 2.05.

An *OMNIDEX-enhanced data base* or *OMNIDEX data base* means a data base enhanced by Omnidex keyword retrieval and/or IMSAM keyed access.

Omnidex, in upper and lower case letters, refers only to the keyword retrieval capabilities that comprise a component of OMNIDEX.

An *Omnidex data base* or *Omnidex-enhanced data base* means a data base enhanced only by Omnidex keyword retrieval.

IMSAM refers to sorted-sequential retrieval and partial-key access.

An *IMSAM-only* data base means a data base enhanced only by IMSAM.

IMSAM intrinsics include the retrieval intrinsics that provide sorted-sequential access and the general intrinsics that perform data base opens, locks, updates and maintenance functions for both IMSAM-enhanced and Omnidex-enhanced data bases. The *Mode* values of IMSAM general intrinsics that correspond exactly to IMAGE *Mode* values are marked with an astersik (*).

A *word* refers to a 16-bit word (2 bytes, or a *half-word* in HPPA usage). Thus a *16 word* array is equal to 32 bytes.

IMAGE refers to the IMAGE/3000, TurboIMAGE/3000 and the TurboIMAGE/XL Data Base Management System provided on HP 3000 and HPPA series computers.

Call conversion refers to the conversion of IMAGE calls to their corresponding IMSAM intrinsics. In earlier editions of the documentation this was called *IMAGE call conversion*. Call conversion is a feature of OMNIDEX and, as such, is supported by DISC.

References to the *DISC account* apply to any of the following accounts:

- DISC205
- DISC206
- DISCTRL
- DISC

Square Brackets

Optional parameters are shown enclosed in square brackets [ ], for example:

```
FO [SETS]
```

Right/Left Angle Brackets

Right and left angle brackets, (<< >>), are used to indicate comments:

```
CALL "DBIEXPLAIN" USING STATUS, PARM.
     <<Note additional parameter!>>
```

Please Note: The screens that serve as illustrations in this manual may not exactly match the screens you may encounter. The function keys and screen size may vary. These screens were captured using Hotshot (Symsoft) while in Reflection (Walker, Richer and Quinn).

Introduction to the Software

OMNIDEX

The OMNIDEX Information Management System (OMNIDEX IMS) is a powerful enhancement to the IMAGE Data Base Management System.

OMNIDEX adds a variety of information retrieval capabilities to an IMAGE data base, using two different types of indexes: Omnidex keyword indexes and IMSAM B-tree indexes. These sophisticated indexes are accessed by OMNIDEX intrinsics that are almost syntactically identical to IMAGE intrinsics.

OMNIDEX provides extremely fast retrievals on any number of fields in an IMAGE data base, including multiple fields in a master set. It also enables you to design data bases that are more logically structured and enables you to develop far more advanced applications than are possible using only IMAGE.

OMNIDEX is used throughout the development of an application to provide a streamlined data base design with state-of-the-art access capabilities. These access capabilities are called *Omnidex keyword retrieval* and IMSAM keyed sequential access.

Omnidex keyword retrieval enables you to instantly locate information using any words or values, or combinations of words and values, as keys. The *keywords* can be in data base records or in external document files. These retrieval capabilities are provided by the Omnidex intrinsics and the keyword indexes.

IMSAM, the IMAGE Sequential Access Method, provides keyed-sequential access to data base records. IMSAM supports partial-key values, and is faster, easier to maintain, more efficient to use and more secure than KSAM. These retrieval capabilities are provided by the IMSAM intrinsics and the B-tree indexes.

The Omnidex and IMSAM intrinsics call the IMAGE intrinsics when accessing a data base, without using Privileged Mode. This means that all the reliability and security features of IMAGE are preserved with OMNIDEX. OMNIDEX will continue to work with future versions of IMAGE. Here is a brief summary of the differences between Omnidex and IMSAM, both of which are part of OMNIDEX:

Omnidex	**IMSAM**
Keyword or partial keyword retrieval.	Partial field retrieval.
Retrieved by search item or Omnidex ID.	Sorted in order by any IMSAM key.
Can easily select across any Omnidex fields within a set.	Can select across fields in a set only in predefined combinations.
Can easily select across related Omnidex sets.	Cannot select across sets.
Uses 3-5 index sets per Omnidex master.	Uses 1 index set per IMSAM key.
Indexes maintained by ODXUTIL.	Indexes maintained by DBIUTIL.
Intrinsic names begin with ODX.	Intrinsic names begin with DBI.

The Omnidex and IMSAM intrinsics can be called by your application programs using any of the following 3GLs: COBOL 74 and 85, FORTRAN 66 and 77, BASIC, SPL, PASCAL, C and RPG.

In addition, the following 4GLs can be directly interfaced with OMNIDEX IMS: PowerHouse (Cognos), Speedware (Infocentre), Insight (Unison Software, Inc.), Synergist (Gateway Systems Corp.), Protos (Protos Software Co.), Today (Computer Power PTY., Ltd.), Rapid (HP) and Fastran (Performance Software).

Among the report writers that interface with OMNIDEX IMS are DataExpress (IMACS Systems Corp.), Easy Reporter (Infocentre), Report Express (Gentry), Visimage (Cogelog) and QUIZ (Cognos).

OMNIDEX IMS retrieval capabilities can even be downloaded to your PC via: DataExpress (IMACS Systems Corp.), Information Access (HP) and OMNIVIEW (DISC).

Note: Specialized documentation is available for developing OMNIDEX applications through many of the languages described above. For more information, contact your DISC sales or technical services representative.

Because OMNIDEX's call conversion library converts IMAGE calls to OMNIDEX calls, most major applications that call IMAGE intrinsics require little or no modification when OMNIDEX has been installed on a data base. Eventual changes are recommended though, to take full advantage of OMNIDEX.

OMNIQUIZ

A separate product called OMNIQUIZ is also available. It provides a direct interface from OMNIDEX to QUIZ, Cognos Corporation's 4GL report writer.

OMNIQUIZ enables you to do Omnidex keyword retrievals and IMSAM sorted-sequential and partial-key retrievals from within QUIZ. It combines the reporting capabilities of QUIZ with the speed and flexibility of Omnidex and IMSAM.

OMNIVIEW

OMNIVIEW is a seamless interface between the familiar environment of Lotus 1-2-3 or Symphony and the data on your HP. It is designed to be used by Lotus 1-2-3 and Symphony users who are familiar with the basics of using spreadsheet software, but who know little or nothing about IMAGE data bases.

OMNIVIEW gives the Lotus 1-2-3 user immediate access to IMAGE data through the data retrieval capabilities of OMNIDEX. OMNIVIEW can:

- ❑ Qualify records based on multiple search criteria in an IMAGE data base.
- ❑ Retrieve on sorted sequential (IMSAM) keys.
- ❑ List records to a spreadsheet.
- ❑ Find the maximum, minimum, average or a sum of numeric fields from a group of qualified records on the HP.
- ❑ Summarize data in cross-tabulated form.

OMNIVIEW imports the results directly into a Lotus 1-2-3 spreadsheet with no manipulation or reformatting required. This enables the creation of work sheets that can reflect current data with little more than a few key strokes.

OMNIWINDOW

OMNIWINDOW is a revolutionary development tool that lets you add the power of Omnidex to your application - without changing a single line of code! In addition to providing OMNIDEX IMS retrieval capabilities, it enables you to design easy-to-use screens, or windows, to a variety of application software. The OMNIWINDOW package includes the OMNILINK data directory software, which enables you to define and restrict access for each user, to particular data bases, data sets and fields.

OMNIWINDOW is easy to implement and maintain. It's easy for your users to learn and use, too. When they want to perform an Omnidex search within an application, they'll simply use a configurable *hot key* combination to invoke the search screens. The access language that is included with OMNIWINDOW enables you to write on-line, context-sensitive help screens, so that there's no learning curve associated with adding OMNIWINDOW capabilities to your application.

Benefits of OMNIDEX

A Full Range of Flexible Retrieval Options

IMSAM provides partial-key access, relational access (>, <, >=, <=, =), and sorted-sequential access to data base records. In addition, Omnidex provides keyword retrieval, partial keyword retrieval, keyword ranges and Boolean combinations of keywords to retrieve data base records or document files.

Omnidex and IMSAM enable any number of fields in any set, including IMAGE masters, to be specified as keys. These keys can easily be specified in an existing data base without restructuring it.

IMSAM keys and Omnidex keyword fields can consist of multiple items or portions of fields and can enable you to retrieve IMAGE details directly without chained reads.

These capabilities enable you to develop more advanced applications than ever before.

Instantaneous Retrieval Speeds

Omnidex and IMSAM retrieve records instantly, even using partial values or combinations of fields.

Omnidex instantly selects those records that contain the specified keywords, even if the selection criteria involve the intersection of several fields. An example would be finding all the customers in California that ordered a certain product last month.

In detail sets, IMSAM keys enable direct access to individual detail records, thus eliminating lengthy chained reads and the high overhead of sorted paths. Also, IMSAM retrieves the records in sorted-sequential order.

An Easy-to-Use, Natural Interface

OMNIDEX enables you to index and retrieve by values in any field you want. This means you can find a customer record, for example, without knowing an account number. All you need is some part of the company name or the contact's first or last name, even the city or the name of an item that customer may have purchased.

In addition, OMNIDEX intrinsics can be called through many programming languages. Therefore, you can call OMNIDEX through a screen formatting language like QUICK, to provide screens and menus for your end users.

Simpler Data Base Designs

OMNIDEX can create multiple keys in an IMAGE master (in addition to the IMAGE search item), enables retrieval across multiple fields, keyed access to details without automatic masters, and sorted-sequential access without external KSAM files.

These capabilities enable you to simplify your data base designs and eliminate many automatic masters and KSAM files. Because the indexes reside in the same IMAGE data base, they are covered by the IMAGE umbrella for security, ILR, logging and other IMAGE utilities.

Faster Application Development

OMNIDEX intrinsics are powerful, high-level intrinsics, compatible with other software like Query, Suprtool from Robelle Consulting, Ltd. and fourth generation languages (4GLs) like QUICK, TRANSACT/3000 and Protos.

These intrinsics are syntactically very similar to the IMAGE intrinsics, so they are easy to learn and incorporate into your programs. For example, the parameters passed to the IMAGE DBPUT intrinsic are the same as those for an IMSAM DBIPUT. The IMSAM DBIPUT intrinsic accepts additional mode values, which are used to perform specialized updates.

Because the Omnidex intrinsics are capable of Boolean and relational logic, programmers benefit by being able to eliminate a substantial amount of IF, THEN, ELSE logic from their program codes.

In addition, the powerful inquiry and support functions of DATADEX can assist in the application development process.

Faster Indexing

The OMNIDEX indexes can be populated at least 20 times faster than IMAGE masters or KSAM files. They are updated automatically every time an entry is added, deleted or modified, so maintenance is minimal.

Components of OMNIDEX

The OMNIDEX Information Management System has 9 major components:

- ❑ DBINSTAL Installation Program
- ❑ ODXPRO Prototyping and Support Program
- ❑ ODXUTIL and DBIUTIL Utility Programs
- ❑ DATADEX Inquiry and Support Program
- ❑ Omnidex Keyword Retrieval Intrinsic Library
- ❑ IMSAM (IMAGE Sequential Access Method) Intrinsic Library
- ❑ Call Conversion and Intrinsic Libraries
- ❑ HPPA Switch Stub Intrinsic Library
- ❑ ODXMGR OMNIDEX-enhanced data base maintenance utility

These components enable the complete implementation and maintenance of a data base application enhanced with OMNIDEX.

Note that the IMSAM keyed sequential access intrinsics and the Omnidex keyword retrieval intrinsics are separate components of OMNIDEX. Omnidex and IMSAM use different index structures to provide their different retrieval capabilities. They also use different utilities to maintain their indexes. Two intrinsic libraries are included with OMNIDEX. One converts IMAGE calls from your application programs into the appropriate OMNIDEX calls. A segmented library handles calls from programs compiled for MPE V, while an executable library handles calls from programs compiled for MPE XL.

Each component of OMNIDEX is briefly described next and described in detail in the chapters that follow.

DBINSTAL Installation Utility

The DBINSTAL installation utility is used to enhance an IMAGE data base with OMNIDEX. It creates all of the necessary data structures to support OMNIDEX. DBINSTAL can be run interactively or in job mode.

DBINSTAL enables you to install, or de-install, and customize the Omnidex and IMSAM retrieval capabilities on your data base. It also enables you to configure OMNIDEX to make the best use of your system resources.

ODXPRO Prototyping and Support Program

ODXPRO is a program provided with OMNIDEX that can generate job files for installing OMNIDEX, populating index sets and for configuring DATADEX. ODXPRO also can generate the default files for interfacing OMNIDEX with QUICK and OMNIQUIZ.

ODXPRO also can create an OMNIDEX prototype from any data set. You can then use DATADEX to perform Omnidex and IMSAM retrievals. This means you can view the retrieval capabilities of Omnidex and IMSAM on your own data almost immediately.

ODXUTIL and DBIUTIL Utility Programs

ODXUTIL and DBIUTIL are the utility programs for OMNIDEX. They are used primarily to populate the index sets after installation. They also provide other diagnostic and maintenance functions.

DATADEX Inquiry and Support Program

DATADEX is a general inquiry and support tool. It can be used as a query utility for standard IMAGE data bases, but it is most useful for the following purposes:

1. **Omnidex and IMSAM Prototypes and Applications**

   An easy-to-use screen processor simplifies Omnidex and IMSAM retrievals and updates of data.

2. **Data Transfer**

   Data set records can be written to MPE files or directly to another data base using the XFER command. MPE files can be used for a report or to load the data into another data base using the BUILD command.

   These functions can help you in implementing OMNIDEX on an existing data base.

3. **Document Management**

   DATADEX supports keyword searches on external text files (ASCII, Editor, HP Word, MPE, Qedit and TDP files). Information about each page, or document, is contained in an IMAGE data set in which each record is called a *catalog* entry. The documents are indexed using the DATADEX KEYWORD command.

   In addition, the DATADEX CONFIG command enables you to configure the IMAGE *catalog* data set to automatically keyword documents when their catalog entries are added.

   After a document file has been indexed, the DATADEX OMNIDEX and VIEW commands can be used to retrieve and view it.

4. **Information about the Data Sets, Keywords and Keys**

   Information about the data base structure can be displayed using the DATADEX FORM command. The Omnidex keyword fields and IMSAM key fields can be listed using the HKW command (help on keywords) and the HK command (help on keys).

5. **Application Testing and Debugging**

   The DATADEX commands can help you test and debug your applications, because they call the Omnidex and IMSAM intrinsics. If a DATADEX command works correctly but your program does not, a bug may exist in your program.

   While more friendly screens can be designed for your end users, DATADEX enables you to configure prompts and displays, enough to satisfy many end users. For more information about DATADEX, see the **DATADEX Reference Guide**.

Omnidex Intrinsic Library

The Omnidex intrinsics are callable procedures used for keyword retrieval and other functions. The Omnidex retrieval intrinsics are:

- ODXFIND
- ODXGET
- ODXGETWORD

The keyword retrieval intrinsics enable you to quickly find the data records that contain the key words or values you specify as retrieval criteria, even with limited information. For example, you no longer need to know a customer number - all you need is the customer name or address. Keyword retrieval enables you to enter:

- Partial words or values, called generic keywords.
- Phonetic or approximate spellings of words.
- Ranges of keywords.
- Combinations of keywords using Boolean operators (AND, OR, NOT).

Omnidex keyword retrieval can be added easily to a new or existing data base to provide a variety of easy-to-use retrieval capabilities. It also can be used to index external document files. The other Omnidex intrinsics are:

❑ ODXINFO	for obtaining information about the data base structure.
❑ ODXPRINT	for generating hard copies of document records.
❑ ODXTRANSFER	for transferring retrieved information to a file.
❑ ODXVIEW	for viewing documents on-line.

IMSAM Intrinsic Library

The IMSAM intrinsics are callable procedures that are used for retrieval and general data base functions. The retrieval intrinsics are:

- ❑ DBIFIND
- ❑ DBIGET

The general intrinsics are:

❑ DBIOPEN	❑ DBILOCK
❑ DBICLOSE	❑ DBIUNLOCK
❑ DBIPUT	❑ DBIERROR
❑ DBIDELETE	❑ DBIEXPLAIN
❑ DBIUPDATE	❑ DBIINFO

The IMSAM intrinsics can be substituted for almost all the IMAGE intrinsics. The exceptions are DBBEGIN, DBEND, DBCONTROL and DBMEMO, which have no IMSAM counterparts.

IMSAM Retrieval Intrinsics

The IMSAM retrieval intrinsics DBIFIND and DBIGET enable sorted-sequential access and partial-key retrieval.

IMSAM sorted-sequential access enables you to retrieve records in sequential order by key value, alphabetically or numerically. The records can be retrieved in either ascending or descending sequential order.

IMSAM partial-key retrieval enables you to specify the first part of a key or a field and retrieve records that begin with that value.

To obtain IMSAM sorted-sequential access or partial-key retrieval, you simply use DBIFIND and, in some cases, DBIFIND with DBIGET with IMSAM modes on an IMSAM field or composite key. The IMSAM modes of DBIFIND support five relational operations: greater than (>), greater than or equal to (>=), equal to (=), less than or equal to (<=), and less than (<). You also can continue to do IMAGE retrievals using standard IMAGE modes for DBIFIND and DBIGET.

IMSAM provides KSAM-like retrieval capabilities, but does so by using indexes within an IMAGE data base. These IMSAM retrieval capabilities can be installed immediately on an existing data base, without modification of the data base and without manually creating index sets.

In addition, IMSAM provides the ability to create new keys, called *composite keys*, from existing fields. A composite key can be part of a field or the combination of several fields concatenated together during IMSAM installation to form one key.

IMSAM General Intrinsics

The general IMSAM intrinsics like DBIOPEN, DBILOCK, DBIUPDATE, DBIPUT and DBIDELETE can replace the IMAGE intrinsics in your programs. These intrinsics work similarly to the IMAGE intrinsics, but they also maintain the data sets used to index IMSAM keys and Omnidex keywords.

For example, the indexes for Omnidex keyword retrieval and IMSAM keyed access are updated quickly and automatically when records are added, modified or deleted using the IMSAM general intrinsics DBIPUT, DBIUPDATE or DBIDELETE. These IMSAM general intrinsics should be used, either directly or via call conversion, whenever a data base enhanced by IMSAM or Omnidex is accessed. This keeps the index sets synchronized with the data sets.

Note that the IMSAM intrinsics call the IMAGE intrinsics to maintain the index sets, so they are always compatible with IMAGE.

Call Conversion Library

The IMAGE call conversion library enables you to optimize your OMNIDEX-enhanced data base in phases. With the call conversion feature, you won't have to modify all the programs accessing a data base immediately after you install Omnidex and IMSAM.

Call conversion traps an application program's calls to IMAGE intrinsics, and converts them into calls to IMSAM general intrinsics. For example, a DBPUT Mode 1 would be converted automatically to a DBIPUT Mode 1, which first adds a record entry, then, automatically updates the Omnidex and IMSAM indexes for you.

Note, however, that you will eventually need to make program changes to make full use of additional IMSAM retrieval modes for partial-key retrieval, and Omnidex intrinsics for keyword retrieval.

The call conversion library also has special logic that enables QUICK and TRANSACT/3000 to access Omnidex and IMSAM intrinsics.

For more information about call conversion, see the **Programming** chapter.

HPPA Switch Stub Intrinsic Library

For those sites using programs that call OMNIDEX Version 2.06 intrinsics in Compatibility Mode, switch stub intrinsic libraries have been provided to enable these programs to make use of the Native Mode OMNIDEX software. They reside in **SL.PUB.DISC**, for the IMSAM and Omnidex intrinsic routines, and **SL.CC.DISC**, for the IMAGE call conversion library routines.

Version 2.05

For those sites using programs that call OMNIDEX Version 2.05 in Native Mode, we have provided switch stub intrinsic libraries for both the IMSAM and Omnidex intrinsic library routines, (**XL.PUB.DISC**) and our IMAGE call conversion library routines (**XL.CC.DISC**).

ODXMGR Maintenance Utility

ODXMGR helps you maintain your Omnidex and IMSAM index sets. It provides commands for data set capacity changes (CAP), detail set reloads (REL) and data set erases (ERA) on the index sets. It also provides commands to list the sets in a data base (FO) or to create a schema from an existing data base (SCH).

Note that ODXMGR has limited command capability; the CAP, ERA and REL commands function only on the Omnidex and IMSAM index sets.

OMNIDEX Implementation

Both the Omnidex and IMSAM retrieval capabilities can be installed on the same data base and even on the same field.

To implement OMNIDEX for the first time on a data base, the primary steps are:

1. **Decide which OMNIDEX version is compatible with your operating system.**

   If you are using the MPE V operating system, you should install OMNIDEX Version 2.05. If you are using the MPE XL operating system, you should install OMNIDEX version 2.06.

2. **Decide which fields you want to index for Omnidex and IMSAM access.**

   See the **Data Base Design** chapter for detailed information about OMNIDEX design. You also can create a simple OMNIDEX prototype of any of your data sets, if you'd like. Such prototypes are useful for measuring performance. See the **Utilities** chapter for detailed information about ODXPRO.

3. **Run DBINSTAL to specify those fields for Omnidex and IMSAM access.** DBINSTAL creates the necessary Omnidex and IMSAM index sets and adds them to the IMAGE data base.

   See the **Installation** chapter for detailed information.

4. **Run ODXPRO and use the JOB INDEX command.** This creates a job stream that runs ODXUTIL and DBIUTIL to populate the index sets. Then, stream the job to load the index sets.

   See the **Utilities** chapter for detailed information.

5. **You can use DATADEX to do Omnidex and IMSAM retrievals on the newly-enhanced data base, and to add, modify or delete data.**

   See the **DATADEX Reference Guide** for detailed information.

8. **Use ODXMGR for data base maintenance on the index sets,** like changing capacities or performing reloads or erasing the index sets.

   See the **Utilities** Chapter for detailed information.

On-line Documentation System

Introduction

An on-line documentation system is provided with OMNIDEX software. This on-line system provides an easy-to-use index for any information about OMNIDEX. The **OMNIDEX IMS Administrator's Guide**, the **OMNIDEX Programmer's Guide** and the **DATADEX Reference Guide** have been indexed using Omnidex, and can be accessed using DATADEX as described in the pages that follow.

To look up information about any topic, access the system and enter words related to the desired topic. You are shown the page numbers where that topic is discussed in the documentation. You also can view those pages on-line.

The on-line documentation system uses the document keywording and retrieval capabilities of Omnidex. This means you can enter any word, combination of words or generic words (partial words with an at-sign (@)) to locate a topic.

How To Begin

First, sign on as MGR.DISC in the DOC group by entering the following at the MPE system (:) prompt:

```
:HELLO MGR.DISC,DOC
```

If passwords have been assigned to the DISC account or to the user MGR, you must obtain the passwords and enter them when you sign on.

Next, enter the ONLINE command:

```
:ONLINE
```

ONLINE is a user defined command (UDC) that runs DATADEX, opens a data base called DOC and sets up function keys on most HP terminals for easy use of the on-line documentation system.

Note that if this UDC has been added to the system UDC, you can use the ONLINE UDC from any account; you need not be in the DISC account.

You will see the DATADEX banner, a welcome message and a screen of information about how to use the function keys. After you press ↵ at this information screen, you will be prompted for a chapter title and document keywords.

Selecting a Chapter And Topic

The first prompt for the on-line documentation system is:

```
Enter Chapter Title (Optional):
```

Note: A general introduction and sample session using the on-line documentation system can be found in Script 2 in the **Demonstration Guide**.

While this **Enter Chapter Title** prompt is displayed, you can press a function key to limit the search to a particular chapter. Function keys F1 through F6 are used to select a particular chapter of the OMNIDEX manual. These function keys and their corresponding chapters are:

Function Key	**Chapter**
F1 - Data Base Design	Data Base Design
F2 - Install	Installation
F3 - Utility	Utilities
F4 - DATADEX	DATADEX
F5 - Intrinsics	Intrinsics
F6 - Programs	Programming

For example, you would press F2 at the **Enter Chapter Title** prompt to limit the search to the **Installation** chapter.

Note: Use ONLINE to find reference topics chapter by chapter or anywhere in the document.

You can enter the chapter title yourself, if you prefer. In fact, you must type **Introduction** or a generic keyword like **Intro@** to select the **Introduction** chapter. You can even select combinations of chapters by entering a keyword list for example, **Design+Intrins@** to select the **Data Base Design** chapter and the **Intrinsics** chapter.

Or you can press ↵ at the **Enter Chapter Title** prompt to find all the pages in the documentation that discuss the topic you want instead of limiting the retrieval to a particular chapter.

After you respond to the **Enter Chapter Title** prompt by selecting a chapter or by pressing $\boxed{\hookleftarrow}$ to select a search on all the chapters, you are prompted for document keywords for the retrieval:

```
Document keywords?
```

This is where you enter words for the topic on which you want information. The more words you enter, the narrower the focus of your search for a particular topic and the fewer pages that qualify. For example, if you are interested in the ODXUTIL SET commands, you could enter the words **SET** and **ODXUTIL** at the **Document keywords** prompt. The number of pages of documentation that qualify would be displayed.

If you first press $\boxed{F3}$ at the **Enter Chapter Title** prompt to select the **Utilities** chapter, then enter **SET,ODXUTIL** at the **Document keywords** prompt, only the pages of the **Utilities** chapter that contain the words **ODXUTIL** and **SET** qualify.

Thus, the more key words you provide on a topic, the more selective the retrieval of the on-line documentation system.

On the other hand, if you select a chapter but don't enter any keywords, the entire chapter can be retrieved, page by page.

If you want help on selecting documentation, you can exit from the **Enter Chapter Title** or **Document keywords** prompts by entering E and then pressing function key $\boxed{F8}$ for **Help**. It displays quick reference information about how to select documentation, as described in this section.

Listing Documents

After you specify a chapter and keywords for retrieval, the number of documents that qualify are shown and you are prompted to list the documents. For example:

```
 5 documents qualify. List?
```

The number of documents is actually the number of pages in the documentation that contain the words you entered. Enter Y for **YES** at the **List** prompt to list the page numbers in the documentation that qualified for the retrieval. When these documentation references are listed, you can view the pages on-line, as described next.

Viewing On-line Documentation

After you list the documents, you are prompted again for a chapter title. You can do another retrieval or you can view the currently qualified on-line documents.

To view documents, first enter E to exit from the **Enter Chapter Title** or **Document keywords** prompt and return to the DATADEX **Command?** prompt. Then, press F7, which is labeled **View Menu**.

Note that F7, like F8 (**Help**), can be used only at the DATADEX **Command?** prompt.

To view on-line documents, use keys F1 through F5 (**View Next**, **View Previous**, **View Keywords**, **Move Forward** and **Move Backward**).

To exit from a document, use F6 (**Exit Document**).

To return to the **Enter Chapter Title** and **Document keywords** prompts, use F7 (**Select Menu**).

These function keys are described in more detail as follows:

Note: F1, F2, F3, F6, F7 and F8 are used at the Command? prompt. F4 and F5 are used to move around within a document.

1. F1 - **View Next**

   Press this function key if you want to view the next page that was qualified by your keyword search. If this is the first key you press, the first page qualified by your keyword search is shown.

2. F2 - **View Previous**

   Press this function key if you want to view the previous page that was qualified by your keyword search. If you press this function key first, the last page qualified by your keyword search is shown.

3. F3 - **View Keywords**

   Press this function key if you want to view the next page that was qualified by your keyword search . **View Keywords** also highlights the keywords entered during the search. If you press this function key first, you see the keywords highlighted on the first page qualified by your keyword search.

Note: To see a description of the View function keys, use F8 (Help).

4. F4 - **Move Forward**

   Use this function key during the actual display of a page of text, not at the **Command?** prompt. It enables you to move forward (down) so you can see the rest of the page.

5. F5 - **Move Backward**

   Use this function key during the actual display of a page of text. It enables you to move backward on the page if you have moved forward.

6. F6 - **Exit Document**

   Use this function key to exit from a page of documentation and return to the **Command?** prompt.

7. F7 - **Select Menu**

   Use this function key at the **Command?** prompt to return to the **Document Selection** portion of the on-line documentation system. The first set of function keys and the **Enter Chapter Title** and **Document keywords** prompts are displayed again, and you can enter keywords to find another topic.

F7 (View Menu) is used only after performing a retrieval, when you want to see the on-line documentation. It displays a new set of function keys for viewing the documentation.

8. F8 - **Help**

   Use this function key at the **Command?** prompt to display descriptions of the current function keys for this **View Menu**.

Viewing Documents by Page Number

After you have viewed a page qualified by keywords then returned to the **Command?** prompt, you can view the next page of on-line documentation. To do so, enter N at the **Command?** prompt to get the next record, then enter V to view it. This is especially useful if you have viewed a page using keyword retrieval and want to view the next page in the documentation.

Note: Enter V at the Command? prompt to view a page of documentation. You can continue using N and V to view subsequent pages.

For example, if a keyword search qualifies page 5-16, you also may want to view the next page, 5-17. Simply enter **N** then **V** to view the next page of documentation. You also can view a specific page of on-line documentation if you know the number of the page you want to see. The command to enter is:

```
FIND PAGE-NO=c-pp
```

where *c* is a chapter number from 1 through 7, and *pp* is a page number from 1 through 999. After you enter the command, the chapter title and page number are displayed.

Exiting the System

To exit from the on-line documentation system, enter **QUIT** at the DATADEX **Command?** prompt.

Maintaining On-line Documentation

The on-line documentation system can be configured, or purged, depending on your users' needs and system resources.

To set up the ONLINE UDC so it can be accessed from any account, do the following:

1. Copy the UDC for ONLINE from UDC.PUB.DISC into the system UDC.
2. Copy the DXUDC file from PUB.DISC to PUB.SYS.

You may want to eliminate the first screen of instructions.

After users are experienced with the system, you can eliminate the first screen of information displayed when you access the on-line documentation. To do this, perform the following steps:

1. Exit from the on-line documentation system.
2. Run the job stream DXUDCFIX.DOC.DISC.

   When the job finishes, access the on-line documentation system and see that the instructions no longer appear.

Purge the documentation files to save space on your system.

If the users do not need to view the on-line documentation, you can purge the data files containing the documentation text to conserve space on your system.

If you purge the documentation files, you can still do keyword selection by document keywords and list the chapters and page numbers that qualify, then look up the pages in the OMNIDEX manual.

Note: This step is not reversible unless you restore the files from the software tape.

The documentation files that you can purge are:

- ❑ DOCINTRO.DOC.DISC
- ❑ DDEX.DOC.DISC
- ❑ DBDESIGN.DOC.DISC
- ❑ INTR.DOC.DISC
- ❑ INST.DOC.DISC
- ❑ PROG.DOC.DISC
- ❑ UTIL.DOC.DISC

Chapter Summary

In this chapter, you learned how to use this manual and the on-line documentation provided with OMNIDEX. You've been briefly acquainted with a few of the powerful information management capabilities that OMNIDEX provides. If you took the time to explore the on-line documentation we've provided, you've had a chance to see, first hand, a little of what OMNIDEX can do.

Training Classes

In the pages that follow, you will learn how to use all of the features of OMNIDEX in your application programs. Although the documentation we've provided will teach you all you need to know, some people benefit the most from hands-on training. For those of you who favor this approach, DISC offers a four day intensive OMNIDEX training class. The classes are taught by members of our technical support staff and are limited to an enrollment capacity of twelve students. Training involves instruction combined with extensive lab work in implementing OMNIDEX's capabilities in the student's own applications. Classes are held monthly at our home office in Denver, Colorado and at other select locations throughout the United States. For more information about OMNIDEX training classes, contact your DISC sales representative.

Chapter Overview

This chapter describes the intrinsics that can be called by your application programs to provide IMSAM and Omnidex retrieval capabilities on an OMNIDEX-enhanced data base.

The first section provides an overview of the intrinsics.

The second section provides detailed information about each intrinsic, including syntax and parameter values.

The third section provides a numbered listing of the calling errors and exceptional conditions that may occur when using the intrinsics.

You also should read the **Programming** chapter of this manual for information about how to call the intrinsics through application programs.

General Intrinsic Information

Introduction

Omnidex and IMSAM intrinsics can be called by an application program to provide enhanced retrieval capabilities on an OMNIDEX-enhanced IMAGE data base.

The intrinsics are divided into two types, Omnidex and IMSAM, because they access two separate indexing structures and do different types of retrievals.

Omnidex intrinsics are called to qualify records based on any combination of keyword values contained in Omnidex fields. They accomplish this by accessing the keyword index sets.

IMSAM intrinsics are called to do sorted-sequential retrievals, based on partial, or full, key values entered for an IMSAM field. They accomplish this by accessing the B-tree index set for each IMSAM key.

Some IMSAM intrinsics are used for both IMSAM and Omnidex to perform puts, updates and deletes. In addition, they automatically perform the corresponding put, update or delete of keys or keywords in the IMSAM and Omnidex index sets. They also provide data base OPEN, CLOSE and LOCK operations.

IMSAM Intrinsics

The IMSAM intrinsics replace, or supplement, most of the IMAGE intrinsics on a one-for-one basis. The IMSAM intrinsics are:

❑ DBICLOSE	❑ DBIINFO
❑ DBIDELETE	❑ DBILOCK
❑ DBIERROR	❑ DBIOPEN
❑ DBIEXPLAIN	❑ DBIPUT
❑ DBIFIND	❑ DBIUNLOCK
❑ DBIGET	❑ DBIUPDATE

These intrinsics use the same number and types of parameters as their corresponding IMAGE intrinsics. They execute identically to their IMAGE counterpart if a standard IMAGE mode value is used. The only differences are:

IMSAM permits several additional values for the *Mode* parameter.

DBIGET and DBIFIND provide IMSAM modes that support generic (partial) specification of the IMSAM key value. The IMSAM key value used in a generic retrieval may range in length from one byte up to the full defined length of the item. The *Mode* parameter can specify the length in words or bytes.

Most IMSAM intrinsics execute identically to their IMAGE counterparts (except for DBIEXPLAIN, see below) if a standard IMAGE mode value is used. This means that a program where IMAGE intrinsics are replaced by IMSAM intrinsics (with no other changes) operates identically to the IMAGE version.

Certain IMSAM and Omnidex intrinsics employ mode options. These are integer values that are added to the *Mode* parameter values to augment the specified mode's operation.

DBIEXPLAIN has an additional parameter called *Parm*, which enables you to configure how errors are handled.

IMSAM and Omnidex intrinsics require a 21 word Status array instead of the 10 word array required in calls to IMAGE intrinsics.

Words 1-10 are still used for the information returned by the IMAGE call, word 11 is the IMSAM or Omnidex condition word, and words 12-21 vary according to the intrinsic being called.

The DBIGET *Dset* parameter is slightly different from the DBGET *Dset* parameter. It can be 16 words (32 bytes) in length, and can include an 8 word (16 byte) value for the IMSAM key name.

Note: Standard IMAGE mode values for IMSAM intrinsics will be marked with an asterisk (*).

Omnidex Intrinsics

The Omnidex intrinsics are:

- ❑ ODXFIND
- ❑ ODXGETWORD
- ❑ ODXGET
- ❑ ODXPRINT
- ❑ ODXINFO
- ❑ ODXVIEW
- ❑ ODXTRANSFER

These intrinsics are similar to IMAGE intrinsics in terms of the parameters that are used. ODXFIND qualifies records based on any combination of keywords (words and values) in one or more fields, and ODXGET retrieves the qualified records' IMAGE search item values. Then, DBFIND and/or DBGET are used to retrieve the qualifying records.

To use the advanced retrieval capabilities of Omnidex and IMSAM, the Omnidex and IMSAM intrinsics must be incorporated into new or existing programs. To review the information about the access capabilities of Omnidex and IMSAM, see the **Data Base Design** chapter of the **OMNIDEX Administrator's Guide.**

Omnidex and IMSAM indexes can be maintained by existing IMAGE applications without program changes. The call conversion library supplied with the software can be used to intercept calls to DBOPEN, DBLOCK, DBPUT, DBUPDATE, DBDELETE and DBCLOSE, and substitute calls to the corresponding IMSAM intrinsics. When the call conversion library is used, the index sets are automatically updated whenever a record is added, deleted or modified (unless deferred updating is in effect).

The call conversion library enables programs like Hewlett-Packard's Query to be used on all IMAGE data bases, including those enhanced by OMNIDEX. In fact, a program may simultaneously access both OMNIDEX and IMAGE-only data bases via the call conversion library.

This means that Omnidex and IMSAM can be used immediately in existing production data base environments. Programs which more efficiently use Omnidex and IMSAM retrieval capabilities can be added gradually to a site's program library.

See **The Conversion of IMAGE Calls** section of the **Programming** chapter for detailed information about using the call conversion library.

IMSAM Keyed Sequential Access

The IMSAM DBIFIND and DBIGET intrinsics provide keyed sequential access and partial-key (generic) retrieval capabilities for fields specified as IMSAM keys.

They replace the IMAGE DBFIND and DBGET intrinsics.

DBIFIND is used for IMSAM keys that are also the search item of a master.

DBIGET is used for all other IMSAM keys.

Generic retrieval means the entries can be accessed without fully specifying the key value. It can be specified by a partial key, starting with the leftmost character.

The *Mode* parameter specifies the length of the key value in bytes or words. It also specifies a relational condition between the key value supplied and the key to be retrieved. The five types of relational retrievals which DBIFIND and DBIGET can perform are:

- Equal to =
- Greater than >
- Greater than or equal to >=
- Less than <
- Less than or equal to <=

For example, a master data set might have a 32-byte IMSAM key for COMPANY. The IMSAM intrinsics would provide keyed sequential access to the master data set via the key and its associated IMSAM B-tree.

This means a master entry can be retrieved by specifying from 1 to 32 characters of a COMPANY name, starting with the leftmost character, and by specifying the relationship (i.e., >, <, etc.) desired between the specified key value and the key sought.

Note that the scope of a relational retrieval is limited to the length specified for the key value. This means that the number of characters that are examined when determining whether an entry satisfies the condition for retrieval depends on the specified length.

For example, **FIND COMPANY = A** would retrieve **ABC** and **ACE**, because the specified key length is only one character (A). But **FIND COMPANY = AC** would retrieve **ACE**, not **ABC**, because **ABC** does not satisfy the criteria; its first two characters do not equal the specified **AC**.

The more characters that are specified, the more precise the retrieval criteria, and the more selective the retrieval.

The key value returned by an IMSAM retrieval is the first value that qualifies for the stated relationship. For *relational operators* (*relops*) =, > and >=, the first value is the first key in ascending sequence, i.e., the lowest key that qualifies. For relops < and <=, the first value is the first key in descending key sequence, i.e., the highest key that qualifies.

For example, a data set might contain several entries with keys starting with L and ranging from **LABEL** to **LUCKY**. A generic retrieval using a relop of = or >= and specifying the character value L would return **LABEL**, while a relop of <= would return **LUCKY**.

Similarly, a relop > retrieval using **L** would return the lowest key starting with **M**, and a relop < retrieval using **L** would return the highest key starting with **K**. The possible returns of relop retrievals are listed:

Relop and Value	**First Key Value Returned**
< L	KUDOS
= L	LABEL
>=L	LABEL
<=L	LUCKY
> L	MAN

Omnidex Keyword Retrieval

The Omnidex ODXFIND and ODXGET intrinsics provide keyword retrieval for each value within a field or group of fields specified for Omnidex keywording.

ODXFIND is used to find which records qualify; ODXGET is used to retrieve the search item values for those records. Then IMAGE Mode 7 or Mode 4 DBGETs or DBFIND followed by Mode 5 DBGETs are used to retrieve the actual records.

Values can be specified for generic retrieval by entering a partial value and appending an at-sign (@), which is used as a wild card. For example, to retrieve records containing values that start with **HEW**, like **Hewlett-Packard**, specify a partial value like **HEW@** for the *Keywords* parameter.

A keyword range also can be specified for retrieval by specifying the lowest and highest values in the range, separated by a colon (:). For example, use **900501:900731** to find all the records with dates from May 1, 1990 to July 31, 1990.

Multiple keywords and Boolean operators can be specified for each Omnidex ODXFIND retrieval. The valid Boolean operators for keyword retrieval are:

- + OR
- , AND
- ,- AND NOT

For example, if you enter the keyword list **USA,HEWLETT,-CA**, the retrieval would qualify records for all the **Hewlett** entries in the **USA**, except those in **California** (provided the keyword fields for company name, state and country were grouped together during installation).

Types of Modes and Mode Options

Most of the IMSAM intrinsics provide three types of modes:

Normal Mode	Affects both the IMAGE data sets and the Omnidex and IMSAM index sets. Used in most cases.
IMAGE-only	Affects only the IMAGE data sets.
Discrete Mode	Affects only the Omnidex and IMSAM index sets.

Normal Mode

Normal mode operations involve a joint effort by IMSAM and IMAGE. For the DBIFIND and DBIGET retrieval intrinsics, a key is first retrieved from an IMSAM B-tree. DBIFIND uses this key to locate a chain head and set up chained access to a detail set. DBIGET uses the key to retrieve a record from the data set.

For the update intrinsics, DBIPUT, DBIDELETE and DBIUPDATE, an entry is first added to, or deleted from, the data set specified by the *Dset* parameter. Then, the corresponding Omnidex index sets and IMSAM B-tree sets are updated automatically.

IMAGE-only

An IMAGE-only mode option is available for the update intrinsics DBIPUT, DBIDELETE, DBILOCK, and DBIUPDATE.

The IMAGE-only mode option is also available for the IMSAM retrieval intrinsics by using a standard IMAGE mode value in the intrinsic. The IMAGE-only mode option bypasses IMSAM and Omnidex, and the IMSAM intrinsic performs identically to its IMAGE counterpart.

IMAGE-only updates enable records to be added to, deleted from and updated in the IMAGE data sets, without updating the corresponding Omnidex index sets and IMSAM B-tree sets. The index sets can be updated later, either by performing a discrete mode update or by performing ODXUTIL and DBIUTIL INDEX operations.

Discrete Mode

The discrete mode option is available for the locking intrinsic DBILOCK, the retrieval intrinsics DBIFIND and DBIGET, and for the update intrinsics DBIPUT and DBIDELETE.

Discrete mode DBIGET is used to retrieve only an IMSAM key, not its corresponding entry. It is useful because it is faster than normal mode. A discrete mode DBIGET need only retrieve the desired key from an IMSAM B-tree; a normal mode DBIGET also must perform a calculated or directed DBGET of the data set entry using the retrieved search item value or record number.

A discrete mode DBIGET can be used whenever only the key value is needed. Discrete mode is faster when DBIGET is used to find a key that matches a specified key value. It generally requires one less disk I/O per value retrieved than normal mode.

If several sequential retrievals are being performed, discrete mode is much faster than normal mode. IMSAM keeps the most recently accessed tree block in the user's stack. Because many keys reside in one physical block, a full block of keys can be retrieved with the same number of disk reads as one key in normal mode.

Discrete mode usually returns keys 20 - 100 times faster than normal mode for sequential retrieval operations. This advantage is greatest for smaller keys because there are more keys per block.

Discrete mode DBIPUTs or DBIDELETEs should be used in two cases:

❑ To correct a discrepancy between the data sets and index sets.

❑ To maintain IMSAM stand-alone B-trees.

Discrete mode DBIPUTs or DBIDELETEs should not be used for other purposes, because they create discrepancies between the data sets and the index sets that contain the Omnidex keyword or IMSAM key values.

In normal mode operations, both the IMAGE data set and its associated indexes are accessed. For example, a normal mode DBIPUT adds an entry to the data set, then adds any Omnidex keyword values contained in the entry to the corresponding Omnidex index sets. It also adds any IMSAM key value(s) to the corresponding IMSAM B-tree (or B-trees if there is more than one IMSAM key field for the entry).

However, a discrete mode DBIPUT adds only the IMSAM key values to the B-trees and the Omnidex keyword values to the Omnidex indexes.

☞ Note: Discrete mode (or normal mode) DBIPUTs must use an at-sign (@) item list. All fields specified in the *Buffer* parameter must be in set definition order.

Stand-alone B-trees can only be accessed via discrete mode because they are associated with dummy masters, which have no entries. Note that a discrete mode DBIDELETE deletes the current key because there is no record to deindex. For more information about stand-alone B-trees, see the **Stand-Alone IMSAM B-trees** section of the **Data Base Design** chapter of the **OMNIDEX Administrator's Guide.**

Omnidex and IMSAM Condition Words

The conventions for *Status* words 1 through 11 are as follows:

***Status* word 1** contains the IMAGE condition word. It is **0** (zero) if an IMSAM or Omnidex intrinsic executes successfully.

***Status* words 1 - 10** contain the Status information returned in the primary or secondary IMAGE call made by an IMSAM or Omnidex intrinsic.

The primary call is the call to the IMAGE intrinsic corresponding to the IMSAM or Omnidex intrinsic being performed. The secondary call is made to update associated IMSAM or Omnidex index sets.

For example, in a normal mode DBIPUT, DBPUT is called first to add the entry passed in the *Buffer* parameter. If this primary DBPUT is successful, various secondary IMAGE calls are then made to update the associated IMSAM B-trees or Omnidex index sets.

***Status* word 11** contains the IMSAM or Omnidex condition word.

If *Status* word 1 (the IMAGE condition word) is zero (**0**), *Status* word 11 (the IMSAM or Omnidex condition word) does not contain a valid error value. These condition word values and their corresponding errors are listed in the **Error Messages** section at the end of this chapter.

If *Status* word 1 is **+888** or **+999**, an IMSAM or Omnidex error has occurred, and *Status* word 11 is also nonzero.

If *Status* word 1 is not zero (0), an error has occurred. Therefore, the IMAGE condition word (word 1) should always be checked for the success or failure of an IMSAM or Omnidex intrinsic call.

There are three categories of errors: IMSAM errors, Omnidex errors and IMAGE call errors.

If an IMSAM error has occurred, *Status* word 1 is set to +999.

If an Omnidex error has occurred, *Status* word 1 is set to +888. In either case, *Status* word 11 is set to the value corresponding to the particular error.

Negative values for word 11 indicate *calling errors*.

Calling errors are essentially syntactical, and result in the inability to execute a call successfully. For example, trying to read an entry in sorted-key sequence by an item that is not an IMSAM key would yield a calling error.

Positive values indicate *exceptional conditions*.

Exceptional conditions are incurred after a call has been executed. They prevent the return of the data requested in the call. Trying to read in key sequence by an IMSAM key past the end of file would yield an exceptional condition error.

IMAGE call errors are errors resulting from a call to an IMAGE intrinsic within an IMSAM or Omnidex intrinsic. If the error occurred in the primary IMAGE call, *Status* words 1 - 10 contain all the IMAGE call information, and word 11 is set to zero (**0**).

For example, referencing a nonexistent data set in the *Dset* parameter or passing a bad item list in the *List* parameter would yield a primary IMAGE call error.

If a secondary IMAGE error does occur, an index set is full, the IMSAM rootfile is damaged, the data base was structurally altered without reinstalling OMNIDEX, or a problem exists in the IMSAM or Omnidex software.

In a secondary call failure, *Status* words 1 - 10 contain the IMAGE condition word resulting from the unsuccessful call, and *Status* word 11 contains an IMSAM internal error code. The contents of *Status* words 2 - 10 depend on the IMSAM or Omnidex intrinsic in question.

To summarize, the IMAGE and IMSAM/Omnidex condition words are:

Status Word 1	Status Word 11	Interpretation
0	0	Successful execution; no errors and no warnings
0	positive	Successful execution with a warning
+888	negative	Omnidex calling error
+888	positive	Omnidex exceptional condition
+999	negative	IMSAM calling error
+999	positive	IMSAM exceptional condition
other non-zero	0	Primary IMAGE call error; words 1-10 contain IMAGE call data
other non-zero	positive	Secondary IMAGE call error; words 1-10 contain IMAGE call data

Each Omnidex or IMSAM intrinsic uses a specific range of condition word values to return exceptional conditions and calling errors in *Status* word 11. Exceptional conditions use positive values and calling errors use negative values.

The values used by each intrinsic are:

Status Word 11	Intrinsic Error
100s	DBIPUT
200s	DBIFIND or ODXFIND
300s	DBIGET, ODXGET or ODXGETWORD
400s	DBIDELETE
500s	DBIINFO or ODXINFO
600s	DBIUPDATE
700s	Miscellaneous
800s	ODXTRANSFER or ODXVIEW
900s	DBIOPEN

A table of condition word values is included with the description of each intrinsic in this chapter. The values and explanations are listed together at the end of this chapter and are updated in a file called ODXINFO.DOC.DISC.

DBIERROR and DBIEXPLAIN can be used to interpret all the listed condition words.

If you receive a value that is not listed, it is an internal error and the DISC Response Center should be called. Internal errors can be caused by a damaged IMSAM rootfile, B-tree or index set.

If the rootfile is damaged, DBINSTAL should be used to reinstall Omnidex and IMSAM and recreate the IMSAM rootfile.

If a B-tree has been damaged, the DBIUTIL INDEX command can be used to rebuild the keys from the associated data set.

If an Omnidex index set is damaged, the ODXUTIL INDEX command can be used to rebuild them.

Intrinsic Definitions

The IMSAM intrinsics are:

DBICLOSE	Closes a data set or data base.
DBIDELETE	Deletes the current entry from a data set.
DBIERROR	Interprets the condition words.
DBIEXPLAIN	Interprets the condition words.
DBIFIND	Locates a chain head before chained DBIGETs.
DBIGET	Retrieves an entry or a key value.
DBIINFO	Provides information about how IMSAM is installed.
DBILOCK	Locks entries, data sets or the data base.
DBIOPEN	Opens a data base and specifies the mode.
DBIPUT	Adds an entry to a data set.
DBIUNLOCK	Releases locks from previous calls to DBILOCK.
DBIUPDATE	Updates an entry in a data set.

The Omnidex intrinsics are:

ODXFIND	Qualifies Omnidex IDs (records or keywords) Based on keywords passed.
ODXGET	Returns the Omnidex SIs of qualifying records.
ODXGETWORD	Returns the next keyword qualified by ODXFIND.
ODXINFO	Returns information about how Omnidex is installed on the data base.
ODXPRINT	Prints a document file.
ODXTRANSFER	Copies the search items qualified by ODXFIND to a file.
ODXVIEW	Displays a document and highlights the keywords.

These intrinsics are described in detail on the following pages.

IMSAM Intrinsics

DBICLOSE

```
DBICLOSE  (Base, Dset, Mode, Status)
```

DBICLOSE terminates an access path to a data base, or closes or rewinds a data set.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See **DBIOPEN** for more information about the base ID.)

Dset Is the name of an array that contains the left-justified name, or the 16-bit number, of the data set being accessed. The data set name may be up to 16 characters long or, if shorter, terminated by a semicolon (;) or a blank (e.g., **CUSTOMERS;** or **ORDER-LINES**).

Mode Is a 16-bit-word integer from 1 to 3. The value corresponds directly to the IMAGE DBCLOSE *mode* value as follows:

- **1** • Terminates access to the data base via a path. The path is identified by the first word of the *Base* parameter, which was assigned by IMAGE during DBIOPEN.
- **2** • Closes the data set specified in the *Dset* parameter.
- **3** • Rewinds (re-initializes) the data set specified in the *Dset* parameter.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

Discussion

Note: DBIOPEN must be used to open an Omnidex or IMSAM data base. Multiple data bases may be open, and each data base may be opened multiple times.

DBICLOSE Modes 2 and 3 are identical to the corresponding DBCLOSE mode values.

DBICLOSE Mode 1 first issues a DBCLOSE Mode 1 to terminate access to the data base via the path specified by the Base ID (the first word in the *Base* parameter). Then the IMSAM and Omnidex local control blocks (ILCB and OLCB) for that access path are deleted. If multiple data bases are accessed serially instead of concurrently, you should close one data base before opening the next to conserve memory resources.

In Version 2.06 the ILCB and OLCB reside in virtual memory of the outer block of the executable library.

Version 2.05

In Version 2.05, the ILCB and OLCB can reside in the DL-DB area, or in an extra data segment when using the call conversion SL or DBIOPEN Mode Option 800.

If the control blocks reside in stack, that space is returned to the system to make it available for future expansion of the stack to accommodate other subsystems or future DBIOPENs If the control blocks reside in an extra data segment, that space is made available for other IMSAM and Omnidex control blocks.

Note: V/3000 also uses the DL-DB area of stack for a forms buffer and other purposes. IMSAM and Omnidex are totally compatible with V/3000 in the way they use DL-DB, despite whether a VOPENFORMF or a DBIOPEN is performed first.

You should close the access paths to all data bases before a program terminates. However, you are not required to close the data bases explicitly because IMAGE is integrated into the MPE operating system. MPE automatically calls DBCLOSE for any data base access paths that are still open when a program terminates.

However, if a program serially performs several DBIOPEN and DBICLOSE calls, you should use VOPENFORMF to open the V/3000 forms file before the first DBIOPEN.

If DBIOPEN is performed before a VOPENFORMF, DBICLOSE is unable to contract the stack after the IMSAM and Omnidex control blocks have been deleted.

Alternatively, an extra data segment may be used for the IMSAM and Omnidex control blocks. This alternative may be selected by using DBIOPEN Mode Option 800 when the data base is opened. This option specifies that all the control blocks reside in an extra data segment.

Note that to open several data bases simultaneously using Mode Option 800, you must specify it for each call to DBIOPEN.

Note: Mode Option 800 should be used with Version 2.05 for PASCAL, FORTRAN 77, COBOL 85 and Business BASIC.

DBIDELETE

```
DBIDELETE  (Base, Dset, Mode, Status)
```

Parameters

DBIDELETE deletes the current entry from the specified master or detail set.

Base Is the name of an array that designates which data base is being accessed. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Dset Is the name of an array that contains the left-justified name, or the 16-bit number, of the data set being accessed. The data set name may be up to 16 characters long or, if shorter, terminated by a semicolon (;) or a blank (e.g., **CUSTOMERS;** or **ORDER-LINES**).

Mode Contains a single, 16-bit-word integer, which corresponds exactly to an IMAGE DBDELETE *mode* value as follows:

1 * Normal mode.

Deletes the current entry from a manual master or detail data set. The IMSAM and Omnidex index sets are updated automatically to remove the IMSAM key values and Omnidex keywords associated with that record.

Mode Options Are integer values that are added to the mode value to elicit the following:

100 **IMAGE-only mode.** The current entry is deleted. IMSAM and Omnidex indexes are not changed.

200 **Discrete mode.** Rereads the current entry and removes the key values and keyword occurrences associated with that record without deleting the entry itself.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

Discussion

If a system failure or program abort occurs while deleting a key or an SI from an index set, the index set may not be updated on disk. It may remain as it was before the attempted key deletion. It is structurally intact, but has an extra key.

In normal mode, DBIDELETE first deletes the desired entry, then performs any necessary operations on the associated index sets. Thus, if the abnormal termination occurred after the entry was deleted but before the index set was modified, a discrepancy would exist. This situation occurs very rarely and is easy to correct by performing a DBIUTIL or an ODXUTIL INDEX or BF (BUILD FAST) operation, as appropriate, on the affected IMSAM key or Omnidex domain.

Note: Subsystem Break (Ctrl -Y) is disabled during a DBIDELETE, and is re-enabled afterwards.

Omnidex and IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

496	ILCB is damaged or OLCB is damaged.
497	Bad base ID, or data base not opened using DBIOPEN.
4xx	Any other 400 level value: IMSAM internal error.

** CALLING ERRORS **

-400	Illegal mode specified.
-402	No current key.
-403	DBIDELETEs of RS indexed details not permitted.

Please see the section on **Error Messages** at the end of this chapter, for more information about these conditions.

DBIERROR

```
DBIERROR  (Status, Buffer, Length)
```

DBIERROR interprets the IMAGE, IMSAM and Omnidex condition words and places an error message in the *Buffer* parameter.

Parameters

Status	Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested. If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.
Buffer	Is a 36-word (72 byte) array where the IMAGE, IMSAM or Omnidex error message is returned.
Length	Is a 16-bit-word integer containing the byte length of the message, up to 72 characters.

Discussion

DBIERROR interprets primary and secondary IMAGE calling errors by calling the IMAGE DBERROR intrinsic internally. All other errors are interpreted directly by DBIERROR.

IMAGE calling errors are errors and exceptional conditions that result from a call to IMAGE within an IMSAM or Omnidex intrinsic. For example, trying to add an entry to a data set that is inaccessible to your user class would result in an IMAGE error. The failing IMSAM intrinsic returns the IMAGE condition word in *Status* word 1. If DBIERROR is called, it uses DBERROR to interpret that error.

IMSAM calling errors and exceptional conditions cause *Status* word 1 to be set to +999. Omnidex calling errors and exceptional conditions cause *Status* word 1 to be set to **+888**. *Status* word 11 is set to the value that identifies the IMSAM or Omnidex error.

For example, a beginning of file error for a descending sequential read would cause an IMSAM exceptional condition; attempting a partial-key retrieval on a non-IMSAM field would result in an IMSAM calling error.

See the section on **IMSAM and Omnidex Condition Words** earlier in this chapter for a discussion of primary and secondary IMAGE calls. A list of IMSAM and Omnidex error codes is included with each intrinsic's description. These error codes are also summarized in the **Error Messages** section at the end of the chapter.

The IMAGE reference manual should be consulted for IMAGE error codes.

DBIEXPLAIN

`DBIEXPLAIN (Status, Parm)`

DBIEXPLAIN interprets the IMAGE, IMSAM and Omnidex condition words and prints an error message on $STDLIST. At the end of the message, the value of *Parm* is displayed after the words, **Program Error**. Note that *Parm* is an additional parameter; it is not a component for IMAGE's DBEXPLAIN.

Parameters

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

Parm Is a 16-bit-word integer parameter. The value of *Parm* determines what happens when there is an error, as listed below.

= 0	The **Program Error** <*Parm*> message is suppressed. Only the IMAGE, IMSAM or Omnidex error message is displayed.
<0	The user process is aborted immediately after the **Program Error** <*Parm*> and error messages are displayed.
>0	The **Program Error** <*Parm*> and error messages are displayed, but the program does not abort.

Discussion

Note that you can use different values for *Parm* on different intrinsic calls when debugging a program. This helps to identify which call is causing problems.

IMSAM and Omnidex calling errors and exceptional conditions are explained by a one or two line textual message. For errors that occur during a primary IMAGE call within an intrinsic, DBIEXPLAIN calls DBEXPLAIN to interpret the error.

Errors that occur during a secondary IMAGE call within an intrinsic are first interpreted using DBEXPLAIN. Then the IMSAM or Omnidex internal error code is printed.

See the section on **IMSAM and Omnidex Condition Words** earlier in this chapter for a discussion of primary and secondary IMAGE calls.

Note: DBIEXPLAIN calls DBEXPLAIN to interpret errors whenever necessary. As noted in the IMAGE reference manual, the *Base*, *Qualifier*, *Dset* and *Password* parameters must be unaltered when DBEXPLAIN is called. Therefore, it is best to call DBIEXPLAIN immediately after returning from the failing intrinsic.

DBIFIND

`DBIFIND (Base, Dset, Mode, Status, Item, Argument)`

DBIFIND locates a chain head in the master set linked to a detail set by an IMSAM search item. The search item is specified by the *Item* parameter. The detail set is specified by the *Dset* parameter.

DBIFIND also sets up pointers in preparation for chained access, and is used only for an IMSAM key that is also a search item in a master set.

A chain consists of detail entries that have the same value, specified in *Item*, as the chain head. The chain head may be located by key value or in sorted-key sequence, based on the *Mode* parameter. The *Argument* parameter contains the full or partial-key value that is used to locate the chain head in directed retrieval modes.

Discrete mode access can be used for all sequential and directed retrieval modes by adding **1000** to the mode value.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Dset Is the name of an array that contains the left-justified name, or the 16-bit number, of the data set being accessed. The data set name may be up to 16 characters long or, if shorter, terminated by a semicolon (;) or a blank (e.g., **CUSTOMERS;** or **ORDER-LINES**).

Mode Is a 16-bit-word integer, as follows:

Directed Modes

1 * Same as IMAGE DBFIND Mode 1 Locates the chain head that exactly matches the full key value in *Argument*.

Additional mode values, to which the number of bytes or words in *Argument* are added, are listed below with their relational operations:

100 Relop = Find. Locates the first chain head in ascending key order whose key value equals the value in *Argument* to the degree specified. For discrete mode, use 1100.

200 **Relop > Find.** Locates the first chain head in ascending key order whose key is greater than the specified key value. For discrete mode, use 1200.

300 **Relop >= Find.** Locates the first chain head in ascending key order whose key is greater than or equal to the specified key value. For discrete mode, use 1300.

400 **Relop < Find.** Locates the first chain head in descending key order whose key is less than the specified key value. For discrete mode, use 1400.

500 **Relop <= Find.** Locates the first chain head in descending key order whose key is less than or equal to the specified key value. For discrete mode, use 1500.

Sequential Modes

These are used only after a call to DBIFIND using a directed mode, to obtain the next key in sequence.

90 Locates the next chain head in ascending key sequence. The *Argument* parameter is ignored.

91 Locates the previous chain head in ascending key sequence (the next chain head in descending key order). The *Argument* parameter is ignored.

92 Locates the current chain head. The *Argument* parameter is ignored.

Mode Options

1000 An integer value of **1000** can be added to the base mode value to specify discrete mode.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is +999, *Status* word 11 contains the IMSAM condition word.

Item Is an 8 16-bit-word (16 byte) array containing the left-justified name of the IMAGE search item, or IMSAM key.

Argument Passes the full, or partial, key value used to locate the chain head for directed retrieval modes. Returns the retrieved key value for discrete modes.

Discussion

In all the directed retrieval modes, the length of the key value supplied for the *Argument* parameter (entered by the user) may vary from 1 byte up to the full defined length of the key. The length of this key value, in bytes or words, must be added to the selected base mode value.

The relational condition selected by the mode is limited in scope to the specified key length. The chain head located is the first key value that satisfies the relation to the precision (length) of the value supplied for the *Argument* parameter.

To specify the precision length of the partial key value in the *Argument* parameter in words, add the number of words to the base mode value. To specify the precision length of the partial key value in the *Argument* parameter in bytes, add the number of bytes to the base mode value and specify the result as a negative integer.

For example, if a two-character key value like **HE** is specified for a find using **>=** as the relop, the mode value should be **301** for the length in words or **-302** for the length in bytes.

If the length of the key value is equal to the defined length of the IMSAM key, that is, a full key value, you need not add words or bytes to the *Mode*.

More examples are included with the DBIGET command description.

Note that if you have an IMSAM key or composite key equal to or greater than 100 bytes, you must use only the base modes or specify the length in words.

If you try to add a partial-key length greater than 100 to a base mode value, it results in the next higher base mode. For example, a base mode of **100** plus a partial-key length of **102** bytes results in a mode of **-202**, which is a > retrieval instead of an =.

Therefore, it is recommended that you use partial-key values on word boundaries if you have extremely large IMSAM keys. For example, a base mode of **100** plus a partial-key value of **51** words results in a mode of **151**, which yields the desired retrieval.

Discrete Mode

Discrete Mode can be specified by adding **1000** to the base mode value. Some examples are: **1090, 1091, 1100, 1300** and **1500**.

A discrete mode DBIFIND differs from a normal mode DBIFIND in two respects:

1. **A discrete mode DBIFIND retrieves the first key value that qualifies based on the *Mode* and *Argument* parameters.** It returns that key value to the calling program via the *Argument* parameter, overwriting the value that was used in the call.

   A normal mode DBIFIND also retrieves the first key value that qualifies, but uses that value in an internal call to DBFIND instead. A normal mode DBIFIND does not alter the *Argument* parameter.

2. **A discrete mode DBIFIND does not internally call DBFIND, but a normal DBIFIND does.** Thus a normal DBIFIND locates a chain head and sets up pointers for chained access (DBIGET Mode 5 or 6) to the detail set, but a discrete mode DBIFIND does not.

Discrete mode DBIFINDs can be used to retrieve all key values that match a partial value entered by a user in an application program. For example, the first DBIFIND does a partial-key retrieval, usually relop =, to retrieve the first key that qualifies. Then sequential mode (1090) DBIFINDs are performed to retrieve subsequent key values in sorted order.

IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

210	Beginning of file.
211	End of file.
213	IMSAM tree is empty.
217	Key not found.
296	ILCB is damaged.
297	Bad base ID, or data base not opened using DBIOPEN.
2xx	Any other 200 level value: IMSAM or Omnidex internal error or IMAGE error.

** CALLING ERRORS **

-200	Illegal Mode specified.
-201	Data set not an IMSAM detail.
-202	Key value exceeds defined key length.
-204	*Item* is not an IMSAM key.
-212	No current key.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

DBIGET

`DBIGET (Base, Dset, Mode, Status, List, Buffer, Argument)`

DBIGET provides several different methods for retrieving all or part of a record via an IMSAM key.

Parameters

DBIGET also can be used to perform IMAGE Mode 4 DBGETs after an ODXGET.

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Dset Is the name of a 16-word (32-byte) array that contains the left-justified name or number of the data set to be accessed and the name or number of the IMSAM key. Note that this differs from the standard IMAGE *Dset* parameter.

The data set name must be in the first 8 words (16 bytes), and the IMSAM key name must begin in word 9 (byte 17). If the data set name is shorter than 8 words, it must be terminated by a semicolon (;) or a blank .

If a master set's search item is the only IMSAM key defined in the set, the last 8 words (16 bytes) are ignored.

If any IMAGE mode is used, the last 8 words (16 bytes) are ignored.

Mode Is a 16-bit-word integer, as follows:

<u>IMAGE Modes</u>

1 * Reread. Standard IMAGE DBGET Mode 1.

2 * Serial read. Standard IMAGE DBGET Mode 2.

3 * Backward serial read. Standard IMAGE DBGET Mode 3.

4 * Directed read. Standard IMAGE DBGET Mode 4.

5 * Chained read. Standard IMAGE DBGET Mode 5.

6 * Backward chained read. Standard IMAGE DBGET Mode 6.

7 * Calculated read. Standard IMAGE DBGET Mode 7.

8 * Primary calculated read. Standard IMAGE DBGET Mode 8.

Directed Retrieval Modes

Base values to which the number of bytes or words are added.

100 Relop = GET. Retrieves the first entry in ascending key order whose key equals the specified key value. For discrete mode, use **1100**. See Note 1 on page .

200 Relop > GET. Retrieves the first entry in ascending key order whose key is greater than the specified key value. For discrete mode, use **1200**.

300 Relop >= GET. Retrieves the first entry in ascending key order whose key is greater than or equal to the specified key value. For discrete mode, use **1300**.

400 Relop < GET. Retrieves the first entry in descending key order whose key is less than the specified key value. For discrete mode, use **1400**.

500 Relop <= GET. Retrieves the first entry in descending key order whose key is less than or equal to the specified key value. For discrete mode, use **1500**.

Sequential Modes

90 Retrieves the next entry in ascending key sequence. The *Argument* parameter is ignored. For discrete mode, use **1090**.

91 Retrieves the previous entry in ascending key sequence (the first entry in descending order). The *Argument* parameter is ignored. For discrete mode, use **1091**.

92 Rereads the current entry. The *Argument* parameter is ignored. For discrete mode, use **1092**.

Mode Options

1000 A value of **1000** can be added to the base mode to specify discrete mode.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+999**, *Status* word 11 contains the IMSAM condition word.

List Is the name of an array that contains an ordered set of data item names or numbers.

Buffer Is the name of the array to which the values of data items specified in the LIST array are moved.

Argument Contains the full or partial key value used to locate the chain head for all directed retrieval modes.

Discussion

DBIGET is used for retrieval on all IMSAM keys, unless you are performing a retrieval on an IMSAM key that is also the IMAGE search item in a master set. If you want to locate a chain head and perform a chained read, you would use DBIFIND.

In all the directed retrieval modes, the length of the key value supplied for the *Argument* parameter (entered by the user) may vary from 1 byte up to the full defined length of the key. The length of this key value, in bytes or words, must be added to the selected base mode value.

Specify the length of the key value in the *Argument* parameter by adding the number of words to the base mode, or adding the number of bytes to the mode value and multiply the result by negative one (-1).

For example, if a two-character key value like **HE** is specified for a find using > as the relop, the *Mode* is **201** for the length in words or **-202** for bytes.

If the length of the key value is equal to the defined length of the IMSAM key, you need not add words or bytes to the *Mode*.

The scope of the relational condition selected is always limited to the specified key length. Some examples are:

- GET the first entry whose key starts with P. Key length is 1 byte and relop is =, therefore: *Mode* = **-101**.
- GET the first entry whose key is greater than **MA** in the first two bytes. Key length is 2 bytes or 1 16-bit-word and relop is >, therefore: *Mode* = **-202** or **201**.
- GET the highest valued key (first in descending order) that is less than or equal to R5A. Key length is 3 bytes and relop is <=, therefore: *Mode* = **-1503** (discrete mode, key only).

  Note that relop <= finds the first key that meets the search argument and displays subsequent keys in descending order. The first key is the highest value in the sequence, for example, a value that is equal to the defined key.
- GET the first entry whose key is greater than **Smith, John Q.**, assuming the defined key length is 14 bytes.

  Key length is 14 bytes or 7 words or 0 (full defined length), and relop is >, therefore *Mode* = **-214** or **207** or **200.**

Note 1: For IMAGE search items that are also IMSAM keys, if the full key value is known and a relop = retrieval is desired, IMAGE Mode 7 should be used instead of Mode 100. Mode 7 (IMAGE calculated access) eliminates a needless B-tree search to locate the key value.

Note 2: If you want to find the highest value for a key, use DBIGET Mode 500 (<=). Don't try a DBIGET Mode 200 (>) with a high value as the argument to reach the end of file, followed by a DBIGET Mode 91 — that retrieves the next-to-last record instead of the last record.

Note 3: You can perform Mode 90 DBIGETs to retrieve all the keys in sequence. However, you must provide a programmatic check to determine when the records retrieved no longer match the specified key value to be retrieved. DBIGET does not automatically check it.

Note 4: For IMSAM keys whose total length is equal to or greater than 100 bytes (including the search item or IMAGE record number), specify partial key lengths on word boundaries. See DBIFIND for more information.

Normal Mode and Discrete Mode

In normal mode, a data set entry is retrieved and the field values specified in the *List* parameter are returned to the calling program.

In discrete mode, DBIGET retrieves only the key values from an IMSAM B-tree. No data set entry is accessed. Discrete mode can be used for all sequential and directed retrieval modes by adding 1000 to the Mode or base mode value.

In discrete mode, the *List* parameter is ignored and the retrieved key is returned to the calling program via the *Buffer* parameter. It has the following format:

Word	Contents of *Buffer* parameter.
1..*n*	Key value, where *n* = key length in words.

If the key is an IMSAM-only key (not also an IMAGE search item), the full key value is returned. The full key is the IMSAM key value plus the record's search item (for a master data set) or the IMAGE record number (for a detail data set).

Only the key is returned for a dummy master with a stand-alone tree.

IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

310	Beginning of file.
311	End of file.
313	IMSAM tree is empty.
317	Key not found.
396	ILCB is damaged.
397	Bad base ID, or data base not opened using DBIOPEN.
3xx	Any other 300 level value: IMSAM or Omnidex internal error or IMAGE error.

** CALLING ERRORS **

-300	Illegal Mode specified.
-301	Data set not an IMSAM data set.
-302	Key value exceeds defined key length.
-304	*Item* is not an IMSAM key.
-305	Image search items require a DBIFIND followed by chained DBIGETs.
-312	No current key.
-315	Not an IMSAM data base.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

DBIINFO

`DBIINFO (Base, Qualifier, Mode, Status, Buffer)`

DBIINFO provides information about how IMSAM is installed on the data base, like what fields were specified for IMSAM keyed retrieval.

Parameters

Base	Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)
Qualifier	Is the name of an 8-word (16 byte) array that contains a data set name or number or an IMSAM key number. In Mode 312, *Qualifier* is a 16-word (32 byte) array, which contains an IMSAM data set name or number followed by an IMSAM key name or number starting at word 9 (byte 17). See the tables which follow.
Mode	Is a 16-bit-word integer that specifies the type of information desired. DBIINFO can be used with the IMAGE modes and functions almost identically to the corresponding IMAGE DBINFO call. There are four (4) additional modes that are used by IMSAM only.

The tables on the following pages list each mode, its purpose, and the contents of the *Qualifier* and *Buffer* parameters.

mode	PURPOSE	*qualifier*	*buffer* ARRAY CONTENTS	COMMENTS
101 *	Defines type of access available for specific item.	data item name or number	element 1 +data item number	If negative, data item can be updated or entry containing it can be added or deleted in at least one data set.
102 *	Describes specific data item.	data item name or number	element 1 . . 8 data item name 9 data type Δ 10 sub-item length 11 sub-item count 12 0 13 0 Note: Δ indicates blank	Left-justified and padded with blanks, if necessary I, J, K, R, U, X, Z, P, CO for Omnidex composite field or CI for IMSAM composite key Integers
103 *	Identifies all data items available in data base and type of access supported.	(ignored)	element 1 *n* 2 ±data item number . . . . *n*+1 ±data item number	*n* = number of data items available Arranged in data item number order. If positive, read-only access. If negative, update or modify access in at least one data set.
104 *	Identifies all data items available in specific data set and type of access supported.	data set name or number	(Same as Mode 103)	(Same as Mode 103 except arranged in order of occurrence in data entry.)

Table 2 - 1: DBIINFO Mode values

mode	PURPOSE	*qualifier*	*buffer* ARRAY CONTENTS	COMMENTS
201 *	Defines type of access available for specific data set.	data set name or number	element 1 ± data item number	If negative, entries can be added or deleted.
202 *	Describes specific data set.	data set name or number	element 1 . . 8 data item name 9 set type △ 10 entry-word length 11 blocking factor 12 IMSAM set type △ 13 0 14 15 number of entries in set 16 17 capacity of set	Left-justified and padded with blanks, if necessary (M, A, D) △ indicates blank integers M – master with IMSAM SI I – data set with IMSAM only keys word integers
203 *	Identifies all data sets available in data base and type of access supported.	(ignored)	element 1 *n* 2 ± data item number . . . . *n*+1 ± data item number	*n* = number of data items available Arranged in data set number order. If positive, read and possibly data item update access. If negative, modify access allowed.
204 *	Identifies all data sets available that contain the specified data item and type of access supported.	data item name or number	(Same as Mode 203)	(Same as Mode 203)

Table 2 - 1: DBIINFO Mode values

mode	PURPOSE	*qualifier*	*buffer* ARRAY CONTENTS		COMMENTS
301 *	Identifies paths defined for specified data set.	data set name or number	element		
			1	*n*	If negative, entries can be added or deleted.
			2	data set number	
			3	search item number	Repeat for each path. If qualifier refers to master, set number is for detail. If qualifier refers to detail, set number is for master. Item numbers identify items in detail.
			4	sort item number	
			. . . . .	. . . . .	
			3*n* −1	data set number	Path designators presented in order of their appearence in the schema.
			3*n*	search item number	
			3*n* +1	sort item number	
			Note:	If sort item is zero, none exists or it is inaccessible. A path designator is not included if user does not have access to search item.	
302 *	Describes specific data set.	master data set name or number	element		
			1	search item number	In master set, zero if inaccessible.
			2	0	
		OR			
		detail data set name	element		
			1	data item number	Of primary path in detail set.
			2	data set number	Of related master set.
					Both are zero if search item is inaccessible.

Table 2 - 1: DBIINFO Mode values

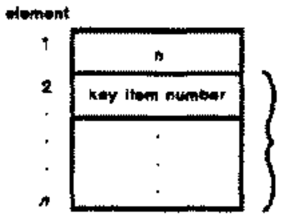
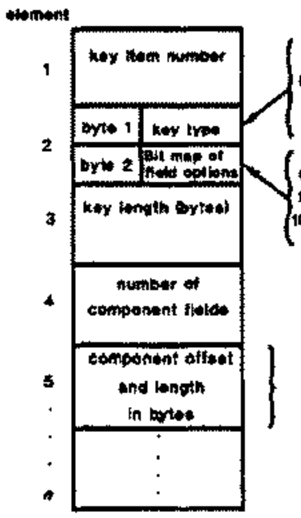
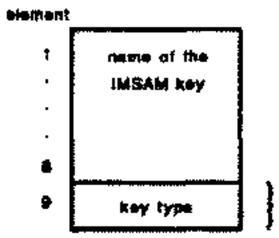
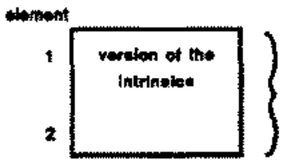
mode	PURPOSE	*qualifier*	*buffer* ARRAY CONTENTS	COMMENTS
311	Identifies IMSAM keys	IMSAM data set name (UPPER CASE) or number (16-bit integer)		*n* = number of IMSAM keys A number greater than 2000 indicates IMSAM composite key
312	Identifies type of IMSAM key, length of key and key components	(32 bytes) IMSAM data set name (UPPER CASE) or number (16-bit integer) IMSAM key name or item number beginning at word 9 (byte 17)	 element 1 key item number 2 byte 1 key type byte 2 Bit map of field options 3 key length (bytes) 4 number of component fields 5 component offset and length in bytes *n*	I = IMSAM only M = SI for manual master A = SI for automatic master 6:1 Batch Indexing 9:1 No Translate 10:1 No Exclude This is repeated for each component in the key
313	Identifies name of IMSAM key,	IMSAM key number		CO for Omnidex composite field CI for IMSAM composite key
321	Identifies version of intrinsics	IMSAM data set name (UPPER CASE) or number (16-bit integer)		e.g., 2.05.03 means version 2 release 5 fix level 3

Table 2 - 1: DBIINFO Mode values

mode	PURPOSE	*qualifier*	*buffer* ARRAY CONTENTS		COMMENTS
401 *	Returns information relating to logging	(ignored)	element		
			1 . . 4	Log Identifier Name	Left-justified and padded with blanks, if necessary.
			5	Data Base Log Flag	1 if data base is enabled for logging, otherwise 0.
			6	User Log Flag	1 if user is logging, otherwise 0.
			7	Transaction Flag	1 if user has a transaction in progress, otherwise 0.
			8 9	User Transaction Number	word
402 *	Returns information relating to ILR	(ignored)	element		
			1	ILR Log Flag	1 if data base is enabled for ILR, otherwise 0.
			2	Calendar Date	Date ILR enabled (mmddyy).
			3 4	Clock Time	Time ILR enabled 1 word (hhmmsstt).
			5	0	Always 0.
			6	▲	▲ indicates blank.
			7 . . 14	▲	Always blank.
			15 16	Reserved	
501 *	Checks subsystem access to the data base	(ignored)	element		
			1	Subsystem Access	0 = no access 1 = read access 3 = read/write access
901 *	Returns MPE code for the Native Language attribute of the data base.	(ignored)	element		TurboIMAGE only
			1	Language ID	

Table 2 - 1: DBIINFO Mode values

Status	Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested. Word 2 contains the number of words of information returned to the *Buffer* parameter. If the call executed successfully, *Status* word 1 is zero (0). If *Status* word 1 is +999, *Status* word 11 contains the IMSAM condition word.
Buffer	Is the name of the array that contains the returned information.

IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

596	ILCB is damaged.
597	Bad base ID, or data base not opened using DBIOPEN.

** CALLING ERRORS **

-501	*Dset* is not an IMSAM data set.
-502	*Item* is not an IMSAM key.
-505	Not an Omnidex keyword field.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

DBILOCK

`DBILOCK (Base, Qualifier, Mode, Status)`

DBILOCK locks entries, data sets or the data base, depending upon the *Qualifier* parameter. Also locks any Omnidex index sets and IMSAM B-trees associated with a data set if one is specified by the *Qualifier*.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Qualifier Is the name of an 8-word (16 byte) array that contains a data set number, a data set name or a set of lock descriptors.

Mode Is a 16-bit-word integer that specifies the type of locking, as follows:

1 * Base level, unconditional locking.

2 * Base level, conditional locking.

3 * Set level, unconditional locking.

4 * Set level, conditional locking.

5 * Item (entry) level, unconditional locking.

6 * Item (entry) level, conditional locking.

When a data set that contains Omnidex keyword fields or IMSAM key fields is locked, or entries within that data set are locked, all associated Omnidex index sets and IMSAM B-trees are also locked.

Mode Options

100 IMAGE-only mode. Locks only the IMAGE data sets, leaving the Omnidex and IMSAM index sets unaffected.

200 Discrete mode. Locks only the Omnidex and IMSAM index sets, leaving the IMAGE data sets unaffected.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

Discussion

A DBILOCK is required before a DBIPUT, DBIDELETE or DBIUPDATE for a data set that contains any IMSAM key fields or Omnidex keyword fields. All associated Omnidex and IMSAM index sets are also locked automatically.

If a set specified in the *Qualifier* parameter is neither an IMSAM nor an Omnidex data set, DBILOCK locks only the set.

Note: If you are updating a detail set that is neither an IMSAM nor an Omnidex data set, and it is linked to an Omnidex automatic master, DBILOCK also locks the automatic master and any details linked to it to insure correspondence between Omnidex IDs.

IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

596	ILCB is damaged.
597	Bad base ID, or data base not opened using DBIOPEN.

** CALLING ERRORS **

-500	Illegal Mode.
-511	Locks on IMSAM or Omnidex index files prohibited.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

DBIOPEN

```
DBIOPEN  (Base, Password, Mode, Status)
```

DBIOPEN initiates access to a data base and establishes the user class number and access mode for all subsequent data base access.

Parameters

Base Is the name of a 16-bit-word array containing a string of ASCII characters. The string must consist of a pair of blanks followed by a left-justified data base name (maximum 6 characters) and terminated by a semicolon (;) or blank ( ), for example,   SALES;. If the data base is successfully opened, IMAGE replaces the pair of blanks with a value called the *base ID*. The *base ID* uniquely identifies this access path between the data base and the process calling DBIOPEN. In all subsequent accesses to the data base the first word of *Base* must be this *base ID*; therefore, the array should not be modified.

Password Is the name of a 16-bit-word array that contains a left-justified string of ASCII characters consisting of an optional password followed by an optional user identifier.

Mode Is a 16-bit-word integer from 1 to 8, identical to IMAGE's DBOPEN modes.

1 • MODIFY with enforced locking. Allow concurrent modify.

2 • UPDATE, allow concurrent update.

3 • MODIFY exclusive.

4 • MODIFY, allow concurrent read.

5 • READ, allow concurrent access in Modes 1 or 5.

6 • READ, allow concurrent access in Modes 6, 2, 4 or 8.

7 • READ, exclusive.

8 • READ, allow concurrent read.

In addition, a mode option may be specified by adding its value to the access mode value. The mode options are as follows:

Mode Options

Added to the base mode value to elicit the following

100 Allows deferred update of the Omnidex indexes. Used with base Mode 3 or 4 only for data base updates, with base Mode 6 for document keywording.

Version 2.05

800 Maintains the Omnidex Local Control Block (OLCB) and the IMSAM Local Control Block (ILCB) in an extra data segment, instead of the DL-DB area of the stack. For example, if Mode Option 800 is desired with an access mode of 1, a DBIOPEN Mode 801 would be used.

If this option is desired for one DBIOPEN call, it must be selected for all other DBIOPENs performed by the same process. See the next page for more information.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

Discussion

When DBIOPEN opens a data base in Modes 1, 2 or 4, it actually opens it twice. This permits OMNIDEX to interrogate data sets without affecting the currency of the primary data base.

The following table illustrates the mode values of both the primary DBIOPEN and the secondary DBIOPEN performed on the data base.

Primary DBIOPEN Mode Value	Secondary DBIOPEN Mode Value
1	5
2	none
3	none*
4	6
5	none*
6	none*
7	none*
8	none*

* Mode 3 is for exclusive access to the data base. See DBIUPDATE for special considerations. Modes 5-8 are read only.

Note: DBIOPEN must be used to open data bases enhanced by OMNIDEX. DBIOPEN first calls DBOPEN to initiate access to the data base. DBIOPEN opens the data base a second time for internal use, but this second OPEN uses negligible system resources and should not affect performance. DBIOPEN also allocates space in the virtual memory of the outer block of the executable library.

DBIOPEN also creates two local control blocks, an ILCB for IMSAM and an OLCB for Omnidex. These local control blocks, which contain information from the IMSAM rootfile, reside in the virtual memory of the outer block of the executable library.

Version 2.05

In Version 2.05, the ILCB and OLCB reside in the DL-DB area of stack. If the data base is opened in Mode Option 800, however, the ILCB and OLCB resides in an extra data segment. Mode 800 must be used for Pascal, FORTRAN 77. COBOL 85 and Business BASIC. See Mode Option 800, discussed earlier, for more details.

Mode Option 800 must be specified in all DBIOPEN calls if multiple DBIOPENs are performed and the option is desired for any of the opens. This is because the control blocks may reside in the stack or in an extra data segment, but not in both places simultaneously.

V/3000 uses the DL-DB area of stack for a forms Buffer and other purposes. IMSAM and Omnidex are compatible with V/3000 in the way they use DL-DB, despite whether a VOPENFORMF or DBIOPEN is performed first. However, if a program serially performs several DBIOPEN and DBICLOSE calls, you should use VOPENFORMF to open the V/3000 forms file before the first DBIOPEN. This allows DBICLOSE to contract the stack after the IMSAM and Omnidex control blocks have been deleted.

By default, word 11 of DL-DB must always be reserved for OMNIDEX if option 800 is used. This reserved word can be changed to another location using the CONFIG command and CHANGELINK option in DBINSTAL. See the System **Resource Considerations** section of the **Appendix** for more information.

Mode Option 100 Deferred Update

If you open a data base using Mode Option 100, the updating of the Omnidex indexes is deferred while you perform adds, modifies or deletes (using DBIPUT, DBIUPDATE or DBIDELETE). IMSAM indexes are still maintained with each call to DBIPUT, DBIUPDATE or DBIDELETE.

Because you can use Mode Option 100 with IMAGE Mode 3 or 4, you can perform a DBIOPEN Mode 103 or 104 to defer updating. You can also defer updating by using the ODXUTIL SET DEFER command, followed by a DBIOPEN Mode 3 or 4. You must use DBICLOSE Mode 1 to close the data base.

If you use Mode 104, other users can access the data base in read-only mode (Mode 6). Still, Mode 103 for exclusive access is recommended, because you can then call IMAGE's DBCONTROL Mode 1 to use deferred puts. This means adding records is much faster than Mode 1 on detail records and on master records in data sets with integer search items.

No other user can access the data base with write access once you begin the deferred update. Also, write access will not be granted to DBIOPEN until the deferred transactions have been posted using the update command in ODXUTIL.

If you need to perform deferred updates on more than one domain, you must first update one domain by performing the ODXUTIL UPDATE. Then, access the data base again to update the next domain by performing another ODXUTIL UPDATE.

Multiple deferred updates can be performed on one Omnidex domain without an intervening ODXUTIL UPDATE. When these are performed, they append to the existing UNLOAD files. Multiple deferred updates can be initiatedvia DBIOPEN Mode 103 or 104, via the ODXUTIL SET command, or a combination of the two.

You can concurrently perform deferred updates on more than one data base in the same group only if you file equate the files that were created by the deferred update.

These files, **ODXUF01** and **ODXUF02**, are created automatically during the deferred update. **ODXUF01** holds the keywords from adds and updates. **ODXUF02** holds the keywords from deletes. Each holds up to two million keywords.

Note: It is possible to fill the **ODXUF01** and **ODXUF02** files if you are updating (adding, deleting or modifying) more than two million keywords. If either of these files become full during the deferred update, the deferred update aborts, and you must reindex the domain using an ODXUTIL INDEX or BF operation.

If you suspect in advance that the update will affect two million keywords or more, equate **ODXUF01** and **ODXUF02** to a bigger file size (i.e., :FILE **ODXUF01;DISC=4000000** to contain four million keywords).

If the ODXUTIL update aborts, you can do one of two things:

- Restore the data base from tape, file equate **ODXUF01** and **ODXUF02** to a larger size and start the deferred update again.
- Perform an ODXUTIL INDEX or BF operation on the domain that was being updated.

To defer the keywording or unkeywording of external documents in a document management application, open the data base in Mode 106. Opening a data base in Mode 106 is useful for indexing document management data bases, because it speeds up the process by a factor of five.

Deferred updates can take longer than an INDEX operation. A INDEX processes one million keywords per hour. A deferred update must sort each ODXUF0# file, copy, erase and sort the XODX' set, merge the records from ODXUF0# with XODX', and add keywords to the ODX' set and the range set.

A deferred update is best if ODX' changes little and the keyword chains are long (i.e., the number of keywords modified is small, but the number of occurrences is large). For example, a deferred update would be useful if you were changing all the occurrences in a STATUS field from N to Y.

A deferred update is slow if you change almost all keywords with a low number of occurrences. For example, a deferred update would be slow if the modifications were unique.

Also, the IMAGE update portion of a deferred update can be slow if you are performing updates on an RS detail set.

Omnidex and IMSAM Condition Word Values

**** EXCEPTIONAL CONDITIONS ****

900	No IMSAM rootfile (IMSAM-ROOTFILE set is empty).
901	Rootfile corrupt.
903	Deferred update must be completed using ODXUTIL.
905	IMSAM has expired (IMSAM error message.) or Omnidex has expired. (Omnidex error message.)
906	Insufficient stack space for ILCB or Insufficient stack space for OLCB.
907	IMSAM-ROOTFILE not accessible (bad password).
908	IMSAM rootfile partially erased.
909	LOADPROC failure. (IMSAM error message.) or MAXDATA too small. (Omnidex error message.)

931	Cannot create file ODXUF0x.
932	Unload file BF/recsize is wrong.
946	Insufficient space in the extra data segment for ILCB.
949	Excluded words list is damaged.
951	Maximum number of extra data segments exceeded.
952	Not enough virtual memory for Omnidex extra data segment.
977	Not a Version 2.05/2.06 rootfile; reinstall IMSAM.
991	DL area contention with another subsystem.
9xx	Any other 900 level value: IMSAM or Omnidex internal error or IMAGE error.

** CALLING ERRORS **

-911	Option 100 requires exclusive update access (Mode 3 or 4).
-912	Mode Option 800 must be elected in the first DBIOPEN.
-914	Call Conversion Library is not a current version.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

DBIPUT

`DBIPUT (Base, Dset, Mode, Status, List, Buffer)`

DBIPUT adds an entry to the specified data set, placing the data that is in the *Buffer* parameter into the fields specified by the *List* parameter.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Dset Is the name of an array that contains the left-justified name, or the 16-bit number, of the data set being accessed. The data set name may be up to 16 characters long or, if shorter, terminated by a semicolon (;) or a blank (e.g., **CUSTOMERS;** or **ORDER-LINES**).

Mode Is a 16-bit-word integer, as follows:

1 * Normal mode. Adds an entry to a manual master or detail data set. Any IMSAM or Omnidex index sets associated with the entry are updated automatically.

Mode Options

An integer value which when added to the base mode elicits the following:

100 IMAGE-only mode. IMSAM and Omnidex indexes are bypassed.

200 Discrete Mode. Indexes the IMSAM key fields and Omnidex keyword fields for an entry without adding the entry itself. An at-sign (@) item list must be used or all fields must be specified in set definition order.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

List	Is the name of an array that contains an ordered set of data item identifiers, which can be names, numbers, or an at-sign (@) for a full list. For any data set that contains Omnidex keyword fields, a full list (@) or the equivalent (all item numbers or names in set definition order) must be used. The only exception to this is that a set in a domain with a transparent ID need not use an @ item list. This means that a set in a domain that uses the TR option or a detail that uses the DR option need not use an @ item list.
Buffer	Is the name of an array that contains a record to be added.

Discussion

DBIPUT automatically assigns the values for an Omnidex ID/SI in sequential order if the field for the Omnidex ID is initialized to zero (**0**) in the *Buffer* parameter of DBIPUT. DBIPUT tries to reuse an Omnidex ID from the free ID list if a negative value, **-1** for example, is used. DBIPUT always assigns the values for the Omnidex ID itself if an alternate field (a field other than the master search item) is used for the Omnidex ID.

If IMAGE-only DBIPUTs are performed, the Omnidex ID/SI is not assigned automatically by DBIPUT, and the next-available ID is not incremented. This allows you to load data that has the Omnidex ID values already assigned. Note that you must then perform an ODXUTIL INDEX or BF to load the values into the indexes.

If a system failure or program abort occurs while you are adding a key to a B-tree, the tree block may not be updated on disk. It may remain as it was before the attempted key addition. It is structurally intact, but it is missing a key.

In normal mode, DBIPUT first adds the desired entry, then performs any necessary operations on the associated index sets. Thus, if the abnormal termination occurred after the entry was added but before the index was modified, a discrepancy would exist.

This situation occurs very rarely and is easy to correct by performing a DBIUTIL or an ODXUTIL INDEX or BF (BUILD FAST) operation, as appropriate, on the affected IMSAM key or Omnidex domain.

Note 1: Subsystem BREAK (Ctrl -Y) is disabled during a DBIPUT and re-enabled afterwards.

Note 2: If the data base was opened in Mode 3 for exclusive access, you may not use * for the item list unless you have already specified an @ item list or the equivalent.

Note 3: Generally, when a key is added to or deleted from a B-tree, only one tree block is affected and the structure of the B-tree is not changed. Only one block is altered. It either has one more key or one less.

Note 4: If discrete puts are used on domains that employ DR indexing, a successful DBIGET must have been performed before the PUT, and the *Status* is left unaffected.

Omnidex and IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

130	Number of tree levels exceeds maximum.
141	IMSAM tree data set is full.
196	ILCB is damaged or OLCB is damaged.
197	Bad base ID, or data base not opened using DBIOPEN.

** CALLING ERRORS **

-100	Illegal Mode specified.
-101	An @ item list is required in discrete mode.
-111	Omnidex ID cannot exceed next available ID.
-112	An @ item list or the equivalent must be used.
-113	Omnidex ID cannot exceed 8,388,607.
-114	Omnidex ID must be greater than 0.
-115	ID cross reference set is full.
-116	ID cross reference set is corrupt.
-117	Maximum chain length for detail with RS indexing is 255.
-119	Deferred updates restricted to one domain.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

DBIUNLOCK

```
DBIUNLOCK  (Base, Dset, Mode, Status)
```

DBIUNLOCK releases all locks set by previous calls to DBILOCK.

Parameters

Base	Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)
Dset	Is ignored.
Mode	Must be a 16-bit-word integer value of 1.
Status	Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested. If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

DBIUPDATE

```
DBIUPDATE  (Base, Dset, Mode, Status, List, Buffer)
```

DBIUPDATE updates the current entry in the specified data set, placing the data that is contained in the *Buffer* array into the fields specified by the *List* parameter.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Dset Is the name of an array that contains the left-justified name, or the 16-bit number, of the data set being accessed. The data set name may be up to 16 characters long or, if shorter, terminated by a semicolon (;) or a blank (e.g., **CUSTOMERS;** or **ORDER-LINES**).

Mode Is a 16-bit-word integer value of 1. It updates the current entry, then updates the IMSAM B-trees and Omnidex index sets as necessary.

Mode Options

A value of 100 can be added to the base *Mode* value to specify IMAGE-only mode. This updates only the IMAGE data sets, leaving the Omnidex and IMSAM index sets unaffected.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888** or **+999**, *Status* word 11 contains the IMSAM or Omnidex condition word.

List Is the name of an array that contains an ordered list of data item identifiers; these can be either names or numbers.

If you use * for the item list, and the data base was opened in Mode 3 for exclusive access, the current list is changed to an @ item list by the update, because a second open could not be performed.

Buffer Is the name of an array that contains the data item values to be added.

Discussion

DBIUPDATE must be used to update data sets containing IMSAM keys or Omnidex keyword fields. It is therefore recommended for updates on all data sets. IMAGE search items cannot be changed by DBIUPDATE.

When the data base is opened in Mode 3, for exclusive access, a DBIUPDATE partial list is upgraded to an @ list automatically to effect the synchronization of the Omnidex and IMSAM index sets.

Note: Subsystem BREAK (Ctrl -Y) is disabled during a DBIUPDATE and re-enabled automatically.

Omnidex and IMSAM Condition Word Values

** EXCEPTIONAL CONDITIONS **

696	ILCB is damaged or OLCB is damaged.
697	Bad base ID, or data base not opened using DBIOPEN.

** CALLING ERRORS **

-611	Alternate ID field cannot be modified.

Please see the section on **Error Messages** at the end of this chapter for more information about these conditions.

Omnidex Intrinsics

ODXFIND

`ODXFIND (Base, Dset, Mode, Status, Field, Keywords)`

Qualifies a number of SIs, IMAGE record numbers, or keywords depending on the way Omnidex has been installed (SI-domain, DR-domain or RS field option), and the *Mode* value used.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Dset Is the name of an array that contains the left-justified name, or the 16-bit number, of the data set being accessed. The data set name may be up to 16 characters long or, if shorter, terminated by a semicolon (;) or a blank (e.g., **CUSTOMERS;** or **ORDER-LINES**).

Mode Is a single, 16-bit-word integer. Modes 1, 2 and 30 are used before an ODXGET; Modes 10 and 11 are used before an ODXGETWORD intrinsic call.

1 Returns the number of qualifying Omnidex ID values.

2 Returns the number of qualifying Omnidex ID values and releases the stack used (recommended with V/3000).

3 Enables the parsing of literal and parenthetical operators in the *Keywords* parameter. When Mode 3 is specified, the keyword list must be terminated by a semi-colon (;).

10 Returns the number of qualifying keywords within the single range or generic value specified and sets a pointer to the first qualifying keyword.

11 Sets a pointer to the first qualifying keyword, as in Mode 10, but does not return a qualifying count.

30 Converts the results of from record specific retrieval to record complex. The list of qualifying IDs is compacted, since multiple occurrences of any SI are reduced to a single occurrence, and record number information is discarded.

Mode Options

To select mode options, add their values, individually or combined, to the mode value. For example, Mode 301 specifies a Mode 1 ODXFIND with Mode Options 100 and 200 enabled.

100	Enables CTRL-Y interrupts during ODXFIND.
200	Disables block mode during ODXFIND.
(300)	Combines Mode Options 100 and 200.
400	Disables the back-out feature on ODXFIND calls that qualify no IDs.
(700)	Combines Mode Options 100, 200 and 400.

Note: Both Mode Options, 100 and 200 (Mode option 300), are recommended with V/3000 to permit $\boxed{\text{Ctrl}}$-Y interrupts.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested. Information is returned as follows:

Word	Contents of *Status* Array
1	IMAGE condition word is zero (**0**).
2-10	Zero (**0**).
11	Omnidex condition word is zero (**0**).
12-13	Number of qualifying IDs (Modes 1, 2 and 30) or number of qualifying keywords (Mode 10).
14-15	Not used.
16-21	Contains the first 12 characters of the keyword that caused a calling error (if the call was unsuccessful because of an incorrect word in the *Keywords* parameter.)

If *Status* word 1 is non-zero, *Status* word 11 contains the Omnidex condition word. DBIERROR or DBIEXPLAIN can be called to interpret the error. See the **Omnidex Condition Word Values** section at the end of this intrinsic description for a list of error numbers and their corresponding error messages.

Field Is the name of an 8 16-bit-word (16 byte) array containing the left-justified name or 16-bit integer item number of a keyword field within *Dset*.

All records that contain the keywords specified in the *Keywords* parameter for this field qualify. If this field was grouped with others during installation, all the records that contain the specified keywords in this field or any other field in the group qualify.

Note: Enter ODX'EXT in this field if you are performing a find on an external document in a document management system. See the **OMNIDEX Document Management Applications** manual for more information about document management systems.

Keywords Is the name of an array containing a list of keywords or partial keywords separated by a plus sign (+), comma (,) or a comma immediately followed by a minus sign (,-).

These symbols represent the Boolean operators: OR (+), AND (,) and AND NOT (,-).

The list of keywords must be terminated by a semicolon (;) or a blank ( ). Blanks embedded in a list terminate the list prematurely, unless they are enclosed in quotes, or Mode 3 has been specified.

Note: The maximum length of the string in the KEYWORD parameter cannot exceed 512 bytes (128 bytes for Mode 3).

Discussion

Modes 1 & 2 are used in preparation for calls to ODXGET. ODXFIND Mode 1 or 2 qualifies record complexes, documents or individual detail records based on a list of keywords, then returns the qualifying count. Record complexes are identified by the Omnidex search item (SI) in the specified data set. Individual detail records are identified by record number if detail record (DR) indexing is used, or by SI value and record number if record specific (RS) indexing is used. Documents are identified by the search item of the associated catalog set. These identifiers are called Omnidex IDs.

Mode 3 is used to parse the *Keywords* list before a search, and the subsequent call to ODXGET. Mode 3 works like Mode 1, but enables a user to refine a search through enhanced parsing logic.

ODXFIND Mode 3 supports parsing operations which enable you to:

- Use literal operators (e.g., **and**, **or**, **not** and **thru**) and spaces (for AND) in a keyword list.
- Override the precedence of Boolean operations through parenthetical nesting (e.g., **(SOFTWARE AND CONSULT@) OR CONTRACT@)**
- Search on JDATE, ASKDATE and PHDATE fields without having to translate keyword search values into the correct date format.

Note: When ODXFIND is called with Mode 3, the *Keyword* list must be terminated by a semicolon (;), because blanks are parsed as an AND operator.

Each of these parsing enhancements are described below.

Literal Operators ODXFIND Mode 3 parses the following strings into equivalent, standard Omnidex operators. Each operation is listed with the current operator in parentheses, and the acceptable literal values. Literal values are not case sensitive.

	Operator	Current Token	Equivalent Tokens
Boolean	and	,	**and** or **#and#** Also, blanks between keywords are parsed as AND operators.
	not	,-	**not** or **#not#**
	or	+	**or** or **#or#**
Range	to	:	**thru** or **..** or **to**
Relational	equal to	n/a	**=** or **eq**
	greater than or equal to	n/a	**>=** or **ge**
	greater than	n/a	**>** or **gt**
	less than or equal to	n/a	**<=** or **le**
	less than	n/a	**<** or **lt**

Parenthetical Operators Parenthetical operators can be used to override the precedence of Boolean operations. For example, all OR operations are performed before AND operations. If you wanted to search for all the software consultants and systems engineers, your keyword list would look like this:

```
(SOFTWARE and CONSULT@) or (SYSTEM and ENGINEER)
```

Without the parentheses, ODXFIND would search for records with the keywords CONSULT@ or SYSTEM, and SOFTWARE and ENGINEER.

Quotes	Quotes can be used around a keyword value to prevent it from being parsed.

Date Fields

A new function, %DATE, supports searches on ASK® and MANMAN® date fields, and PowerHouse JDATE and PHDATE date fields. %DATE can be entered as a keyword in a keyword list using the following syntax:

```
%DATE(date,order,type)
```

Each of the parameters are discussed below:

date	This represents the date value on which you are searching. For example, **900509** or **050990.**
order	This represents the order of the *date value.* The acceptable values for *order* are: **YYMMDD**, **MMDDYY** and **DDMMYY.** If no value is entered, *order* defaults to YYMMDD. This parameter is positional and must be delimited by commas, as in the example below.
type	This represents the type of date field on which you are searching. Acceptable values are: **ASKDATE**, **PHDATE** and **JDATE** .

A sample search using the %DATE function might be:

```
TRANS-DATE?   %DATE(890930,,ASKDATE):%DATE(900101,,ASKDATE)
```

The *order* was not specified because the date values were in the YYMMDD default order. Because the TRANS-DATE field is in ASKDATE format, the *type* was specified as such. The entire %DATE expression is treated as a keyword, which is why there is a %DATE expression on either side of the range operator.

In **Mode 10 or 11**, the list must contain only one partial keyword or a keyword range immediately followed by a semicolon (;) or blank . This partial keyword or keyword range defines a list of keywords that may be retrieved one at a time using ODXGETWORD.

ODXFIND Mode 10 or 11 finds all the keywords that match a partial keyword value or fall within the range of two keyword values. These modes are used before retrieval of those keywords using ODXGETWORD. In Mode 10, the number of qualifying keywords contained within the generic/range specification is also returned.

For example, DATADEX calls a Mode 10 ODXFIND when a user enters a exclamation point (!) followed by a partial keyword or a keyword range at a prompt for Omnidex keywords.

ODXFIND Mode 30 can be used after a record specific retrieval on an RS keyword field to compact the list of record numbers into a list of Omnidex SI values, effectively converting individual records into record complexes.

The *Keywords* parameter supports the following operations:

Ranges

A keyword range may be specified by entering a starting value, a colon and an ending value. Either the starting value or the stopping value is optional.

The syntax is:

[*Start*] : [*Stop*]

Start:Stop	Means all keywords from **Start** through **Stop**
Start:	Means all keywords greater than or equal to **Start**
:Stop	Means all keywords less than or equal to Stop.

When Mode 3 is used, the operators **thru**, **..** and **to** can be used in place of the colon.

If the *Start* and *Stop* values are numbers and the field is a character field (IMAGE types U, X and Z), they must contain the same number of digits. A numeric ordering is used to determine the list.

If either value is non-numeric, the values need not contain the same number of characters, and an alphanumeric ordering is used. For example, ASCII values like **123**, **12BA**, **12/22**, **123.5** and **12.3** fall within the alphanumeric range **110:130A**, but only **123** and **123.5** fall within the numeric range **110:130.**

ASCII ranges also can be generic (partially specified), by appending an at-sign (@) to a partial *Start* or *Stop* value.

ASCII ranges must have the same number of characters in the start and stop values. A range of **130:3000** would return the error message **Start and stop values must have the same number of digits.** The range could, however, be specified as **130:999+1000:3000.**

Relational Operations

Relational operations can be performed either by calling ODXFIND in Mode 3, or by combining open-ended range operations with Boolean operations.

In Mode 3 calls, any of the following relational operations can be performed using one of the tokens listed:

equal to	= or **eq**.
greater than	> or **gt**
greater than or equal to	>= or **ge**
less than	< or **lt**
less than or equal to	<= or **le**

Relational operations also can be performed by combining open-ended range operations with Boolean operations using the following syntax. Note that *value* represents a keyword value:

equal to	*value*
greater than	*value*:,-*value*
greater than or equal to	*value*:
less than	:*value*,-*value*
less than or equal to	:*value*

Generic

Keywords also may be generic (partially specified), by appending an at-sign (@) to a partial keyword. @ is a wild card, so a generic keyword covers all the words or values that begin with the same characters. For example, **MANAG@** includes MANAGE, MANAGER, MANAGING, etc.

Note: Generic values are not supported for searches on binary fields

Samelist

Records from prior retrievals can be qualified by using either of two operations. The simplest is the Samelist operator (*). After an ODXFIND has been performed, you can use the Samelist operator to start with the most recently qualified list of IDs. You can then further qualify the list, add to it or subtract from it by specifying additional keywords.

Samelist AND (*,) qualifies the list further based on the additional keywords. **Samelist OR** (*+) adds the IDs qualified by the additional keywords to the list. **Samelist NOT** (*-) limits the list by subtracting the IDs qualified by the additional keywords.

External ID files The second operation employs external ID files created by ODXTRANSFER. To select records using IDs qualified from one or more previous ODXFINDS, including selections in another data base, ODXTRANSFER Mode 201 must first have been used to transfer the IDs to an external file. The file can then be specified in an ODXFIND by placing the file name, preceded by $, in the keyword list. The file name is a functional replacement for a keyword and follows the same syntactic rules. ODXTRANSFER is discussed later in this chapter.

Soundex The Soundex operator (!) is used to specify a phonetic, or Soundex, search on a designated Soundex field by appending it to the keyword(s) in question. When used on any field other than a Soundex field, or a group where the Soundex field is not specified as the first item in the *Itemlist*, error message -224, **Not a Soundex Field**, is returned.

☞ Note: If a Soundex field is grouped with other fields, be sure to specify the Soundex field name when calling ODXFIND for a Soundex retrieval.

Multifind Multifind operations are used to continue ODXFIND searches across Omnidex domains, even if the target domain is in another data base.

In Multifind, keyword values from records that qualified in a previous ODXFIND can be used to qualify records in a second, *target*, domain. The keyword values used in a Multifind search can be supplied either from memory, or from an external file.

For a Multifind search to use values supplied from memory, they must be SI values returned from a previous ODXFIND. When a Multifind is using SI values returned from a previous ODXFIND, an ampersand (**&**) in the *Keywords* parameter is all that is required to supply the SI values from memory.

When Multifind uses search values that have been written to an external file, the file is passed through the *Keywords* parameter preceded by an ampersand (**&**). For example, the *Keywords* parameter would contain **&*filename***, where *filename* represents the name of the external file containing the search values. For more information about creating files for Multifind searches, see the **ODXTRANSFER** section of this chapter, or the **XFER OMNIDEX** command in the **DATADEX** manual.

Multifind requires that the *target* field in the target data set is an Omnidex key. The name of the target field for a Multifind search is passed via the *Field* parameter, as in any ODXFIND.

The Multifind operator (**&**) must be specified as the first character in the *Keywords* parameter. Any keywords or operations that follow a Multifind operation (& or *&filename*) are ignored.

Version 2.05

ODXFIND requires stack space to manipulate the qualified IDs during the Omnidex retrieval. For information on stack size considerations, see the **Programming Considerations** section of the **Programming** chapter, and the **Appendix** of this guide.

See the OMNIDEX command in the **DATADEX Reference Guide** for more information about ODXFIND retrieval operations.

If a keyword in the keyword list does not exist for the specified Omnidex field or group, no entries qualify unless the non-existent keyword is combined with other keywords that do exist via an OR operation. Otherwise, error 217, **No entries contain** ***<keyword>*** is returned.

Even if all keywords in the list exist, no entries may qualify, depending on the Boolean conditions specified in the list. For example, if both A and B exist, but not in the same record or record complex, no entries are selected. Here, *Status* word 1 would be set to zero (**0**) to indicate a successful call, while *Status* words 12-13 would return a qualifying count of **0**. This is consistent with an IMAGE DBFIND, which succeeds if the specified search item exists, even if the chain count is zero (**0**).

This has implications for error checking in programs. Following an ODXFIND, your program must check for a non-zero *Status* word and perform error processing. It also must check for a qualifying count greater than zero (**0**). If the qualifying count equals zero (**0**), it should display a message saying **No entries qualify** and re-prompt for keywords. See the **COBODXS.DEMO.DISC** program for an example of error processing.

When a keyword list returns a qualifying count of zero (**0**), ODXFIND automatically backs out to the previous list of qualifying IDs. An interactive user can then try different selection criteria, or skip the field and proceed to another field, if a multi-field selection is being done.

In batch mode, however, reprompting is not possible, and a failed selection should generally not be backed out. Mode Option 400 may be specified to disable the back-out feature.

Omnidex Condition Word Values

** EXCEPTIONAL CONDITIONS **

201	CTRL-Y interrupt.
212	Stack overflow: Control buffers.
213	Too many keywords in list.
217	No entries contain keyword.
296	OLCB is damaged.
297	Bad base ID, or data base not opened using DBIOPEN

** CALLING ERRORS **

-201	Keyword is an excluded word.
-202	Illegal ODXFIND mode.
-203	Data set is not an Omnidex set.
-204	Illegal use of NOT operator.
-205	Bad keyword list.
-207	Keywords must start with an alphanumeric character or decimal point.
-208	Start and stop values must have the same number of digits.
-209	Bad item name specified.
-210	Item is not an Omnidex keyword field.
-211	Only one generic keyword or one keyword range permitted.
-212	List must contain a generic keyword or keyword range.
-213	Samelist not allowed in discrete modes (10 and 11).
-214	Not an Omnidex database.
-216	Generic keywords not allowed on binary fields.
-217	Keyword contains a non-numeric character.
-218	Missing keyword in the list.
-219	Unmatched quote.

-220	Missing terminator (semicolon or blank).
-221	Start and stop values must have the same sign.
-222	Split retrievals not allowed with Samelist. OR
-223	*Dset* must be the same as in previous RS retrieval.
-224	Not a Soundex field.
-231	Not an ODXTRANSFER ID file.
-232	Multifind cannot reference an ODXTRANSFER ID file.
-233	File record size exceeds max keyword size.

Please see the section on error messages at the end of this chapter for more information about these conditions.

ODXGET

`ODXGET (Base, Mode, Status, SI-List, SI-Count)`

ODXGET moves one or more Omnidex SIs (search items) or record numbers into the *SI-List* parameter in preparation for a DBGET.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Mode Is a single, 16-bit-word integer (0 - 4) that indicates the type of retrieval and the pointer movement.

0 Rewind. Resets the pointer. The pointer is reset to the beginning of the list if the next operation is a forward read or spacing (Mode 1 or 3). The pointer is set at the end of the list if the next operation is a backward read or spacing (Modes 2 or 4). If the previous ODXFIND was performed on a record specific field, *Status* word 16 contains the value -1. Otherwise it contains 0.

1 Forward read. Reads the next *SI-Count* SIs, or IMAGE record numbers (R#s) for DR detail sets, into the *SI-List* array, and updates the internal pointer.

2 Backward read. Reads the previous *SI-Count* SIs (or R#s) into the *SI-List* array and updates the internal pointer.

3 Space forward. Moves the pointer forward by the integer count specified in *SI-Count*. Does not return any SIs or R#s.

4 Space backward. Moves the pointer backward by the count specified in *SI-Count*. Does not return any SIs or R#s.

Mode Options

In addition, there are mode options that are used for record specific (RS) retrievals. They are specified by adding the mode option value to the base mode value. For example, ODXGET Mode 11 performs a forward read for a retrieval after an ODXFIND on an RS field in a detail set.

10 Returns the record number instead of the search item. Applies to RS keyword fields in detail sets only.

20 Returns the search item and the IMAGE record number. Applies to RS keyword fields in detail sets only.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**), and the contents of the *Status* array are:

Word	Contents of *Status* array
1	IMAGE Condition word is zero (**0**).
2-10	Not used.
11	Omnidex condition word is zero (**0**).
12-13	Number of SIs or record numbers returned in *SI-List* for Modes 1-4. Number of qualifying IDs for Mode **0**.
14	Search item length, in 16-bit-words.
15	Current Omnidex data set number.
16	Contains the value -1 if the previous ODXFIND was performed on a record specific detail field; otherwise contains zero (**0**).
17-21	Not used.

If *Status* word 1 is **+888**, *Status* word 11 contains the Omnidex condition word. Call DBIERROR or DBIEXPLAIN to interpret the error. See the **Omnidex Condition Word Values** section at the end of this intrinsic description for a list of error numbers and their corresponding error conditions.

SI-List Is the name of an array into which the requested IMAGE SIs are to be moved. The array must be defined as binary or ASCII, depending on the data type of SIs to be returned, and large enough to hold the number of SIs specified by *SI-Count.*

SI-Count Is a single, 16-bit-word integer count of the number of SIs that are to be moved into the SI-List. The allowed range is from 1 to **4096**. If *SI-Count* exceeds the number of IDs remaining in the Omnidex internal list from ODXFIND, only the remaining SIs are returned. If no IDs are left to be moved, an error is returned. In any case, *Status* words 12-13 contain the number of SIs moved.

In Mode 3 or 4, *SI-Count* specifies the number of SI positions the pointer is to be moved forward or backward. This number can be between **1** and **32,767**. If the requested *SI-Count* would move the pointer to a position before the beginning, or past the end, of the file, an error is returned and the pointer's position is left unchanged.

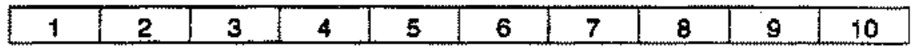
Discussion

An ODXFIND, which qualifies Omnidex IDs, must be performed before calling ODXGET. The Omnidex ID represents a search item (SI) value after a selection on an SI-domain, a search item and record number after a selection on a detail set using record specific (RS) indexing, or a record number after a selection on a detail set using Detail Record (DR) indexing.

Each time ODXGET is called, Omnidex moves an internal pointer for the list of qualifying IDs. This pointer can move forward or backward, or be reset to the beginning of the ID list.

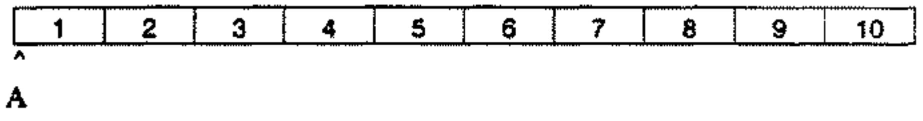
The SI or record number values, which correspond to the Omnidex ID values, are used to retrieve records using standard IMAGE DBGETs for master records, or DBFIND plus DBGET for detail records.

For example, you may have qualified 10 IDs using ODXFIND:

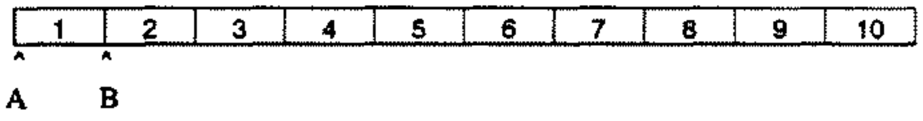


After qualifying 10 IDs, the following examples of ODXGET retrievals would retrieve SIs and place the pointer in the marked positions.

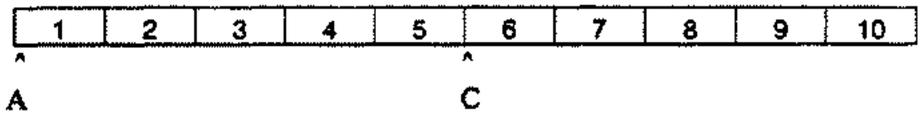
A. Immediately after ODXFIND, the pointer would be in the first position (point A):



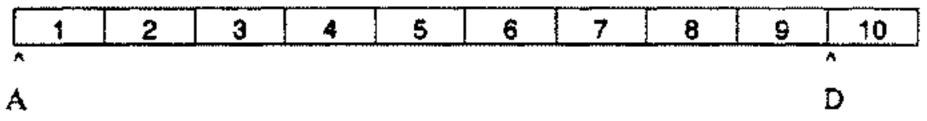
B. Starting at point A, an ODXGET Mode 1 with an *SI-Count* of 1 would retrieve SI number 1 and would set the pointer at point B:



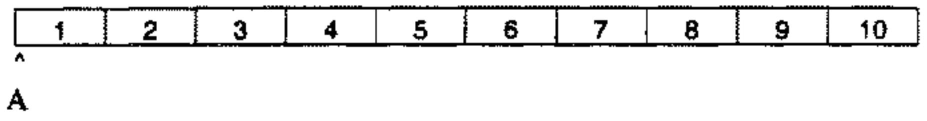
C. Starting at point A, an ODXGET Mode 1 with an *SI-Count* of 5 would retrieve SIs 1 - 5 and would set the pointer at point C:



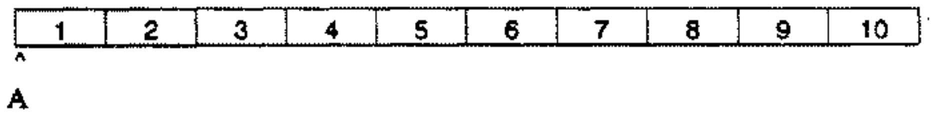
D. Starting at point **A**, an ODXGET Mode 2 with an *SI-Count* of **1** would retrieve SI number 10 and would set the pointer at point **D**:



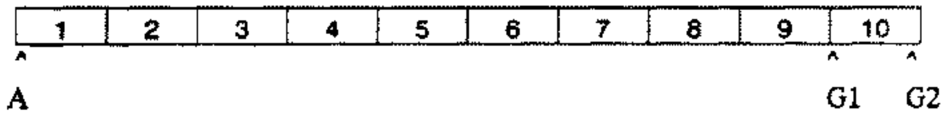
E. Starting at any point, an ODXGET Mode 0 resets the pointer to point **A**:



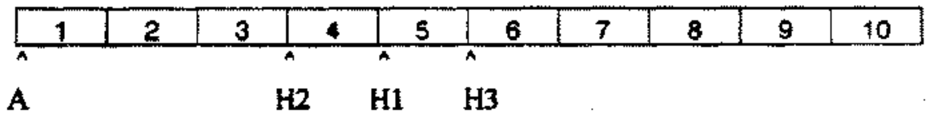
F. Starting at point **A**, an ODXGET Mode 3 with an *SI-Count* of **10** would return an exceptional condition error, **End of ID List**, and would leave the pointer at point **A**:



G. Starting at point **A**, an ODXGET Mode 3 with an *SI-Count* of **9** would not retrieve any SIs and would set the pointer at point **G1**. A subsequent ODXGET Mode 1 with an *SI-Count* of **1** would retrieve SI number 10 and would set the pointer at point **G2**:



H. Starting at point **A**, an ODXGET Mode 4 with an *SI-Count* of **6** would return no SIs and set the pointer at point **H1**. A subsequent ODXGET Mode 2 with an *SI-Count* of 1 would retrieve SI number 4 and would set the pointer at point **H2**:



If, instead of a Mode 2, a Mode 1 were used, SI number 5 would be retrieved and the pointer would be set at H3.

For Omnidex retrievals on a DR (detail record indexed) set, ODXGET returns the record number instead of the SI value. The record number is then used with DBGET Mode 4 to perform the retrievals.

For Omnidex retrievals on RS (record specific) keyword fields in a detail set, ODXGET Mode Option 10 or 20 may be added to the base mode of 1 through 4. Mode Option 10 returns the record number, instead of the SI value, for the qualifying detail record, which enables you to use an IMAGE DBGET Mode 4 to retrieve the record. Mode Option 20 returns both the SI value and the record number.

Omnidex Condition Word Values

** EXCEPTIONAL CONDITIONS **

310	Beginning of ID list.
311	End of ID list.
396	OLCB is damaged.
397	Bad base ID, or data base not opened using DBIOPEN.

** CALLING ERRORS **

-300	Illegal mode specified.
-301	Bounds violation: Target (Count).
-303	No preceding ODXFIND.
-305	Preceding ODXFIND was not record specific.
-314	Not an Omnidex database.
-315	Not an IMSAM database.

Please see the section on error messages at the end of this chapter for more information about these conditions.

ODXGETWORD

`ODXGETWORD  (Base, Mode, Status, Target)`

ODXGETWORD reads the next keyword in the list qualified in the last Mode 10 or 11 ODXFIND. This allows users to see the keyword choices available to them before they enter those keywords to qualify records.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Mode Is a single, 16-bit-word integer (1 or 2) as follows:

1 Returns the next keyword in ascending sequential order.

2 Returns the next keyword in ascending sequence and the number of entries that contain the keyword.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested. If the call executed successfully, *Status* word 1 is zero (**0**) and the contents of the array are:

Word	Contents of *Status* array
1	Condition word is zero (**0**).
2-10	Not used.
11	Omnidex condition word is zero (**0**).
12-13	For Mode 2, these words contain the number of entries that contain the specified keyword.
14-15	Double-word zero.
16-21	Not used.

Target Is the name of an array into which the next keyword in the range is to be moved. The array size must be 24 bytes.

Discussion

Note that ODXGETWORD automatically converts binary field data into ASCII characters.

ODXFIND is first called using Mode 10 or 11. The *Keywords* parameter must contain one partially specified keyword or a keyword range.

ODXFIND Mode 10 returns the number of qualifying keywords in the specified range and sets an internal pointer to the first word in the list. Mode 11 does not return a count, but sets the pointer to the first word in the list.

ODXGETWORD is then called to move one keyword into the target array. ODXGETWORD can be called repeatedly to retrieve all the keywords in the list. After the last qualifying keyword is reached, an **End of Range** condition is returned.

Omnidex Condition Word Values

** EXCEPTIONAL CONDITIONS **

314	End of Range.

** CALLING ERRORS **

-300	Illegal mode specified.
-302	No preceding mode 10/11 ODXFIND.

Please see the section about error messages at the end of this chapter for more information about these conditions.

ODXINFO

`ODXINFO (Base, Qualifier, Mode, Status, Info)`

ODXINFO returns information about how Omnidex is installed on the data base: what fields are keyworded, how fields are grouped, etc.

Parameters

Base	Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)
Qualifier	Is the name of an 8 16-bit-word (16 byte) array that contains a data set name or a single, 16-bit-word integer data set number for which information is requested. In Mode 6, the *Qualifier* array also contains an integer group number in word 9 (byte 17).
Mode	Is a single, 16-bit-word integer value that specifies what type of information is desired. The modes, the type of *Qualifier* necessary for each mode, and the information returned to the *Info* parameter are shown in the tables that follow.

mode	PURPOSE	*qualifier*	*Info* ARRAY CONTENTS		COMMENTS
			element		
1	Identifies keyword fields for a set	Omnidex data set name or number	1	data set number	
			2	set number of Omnidex master of the set	
			3	item number of Omnidex SI for the set	
			4	offset (in words) of Omnidex SI for the set	Contains 0 if the set is a Detail Record indexed set.
			5	number of keyword fields in the set	
			6	item number of each keyword field in the set	The size of this array depends on the number of keyword fields in the set (Word 5). It must hold one 16-bit word for each keyword field in the set.
			.	.	
			.	.	
			.	.	
			n		
			element		
2	Identifies keyword grouping for a set	Omnidex data set name or number	1	data set number	
			2	set number of Omnidex master of the set	
			3	item number of Omnidex SI for the set	
			4	offset (in words) of Omnidex SI for the set	Contains 0 if the set is a Detail Record indexed set.
			5	number of keyword fields in the set	
			6	Word 1: item number of a keyword field; Word 2: group number of a keyword field	The size of this array depends on the number of keyword fields in the set (Word 5). It must hold one 16-bit word for each keyword field in the set. Zero in word two means the field is ungrouped.
			.	.	
			.	.	
			.	.	
			n		
			element		
3	Identifies the Omnidex sets in the domain of the specified master set.	Omnidex master set name or number	1	master set number	
			2	number of Omnidex sets in this domain	Includes any Omnidex manual master sets.
			3	number of each set in this domain	The size of this array depends on the number of Omnidex data sets (Word 5), up to a maximum of 17 sets (including the specified Omnidex master).
			.	.	
			.	.	
			.	.	
			n		

Table 2 - 2: ODXINFO Mode values

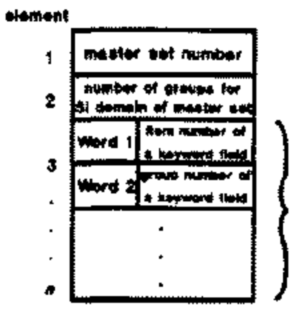
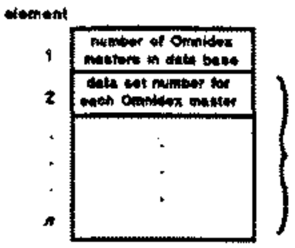
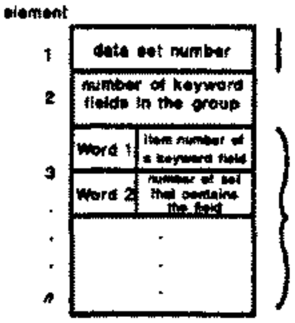
mode	PURPOSE	*qualifier*	*info* ARRAY CONTENTS	COMMENTS
4	Identifies keyword grouping for the domain of the master specified in qualifier.	Omnidex set name (UPPER CASE) or number	 element 1 master set number 2 number of groups for SI domain of master set 3 Word 1 item number of a keyword field Word 2 group number of a keyword field *n*	The size of this array depends on the number of groups (Word 2). It must be large enough to contain two 16-bit words for each group, up to a maximum of 63 groups.
5	Identifies the Omnidex master sets for a data base	(ignored)	 element 1 number of Omnidex masters in data base 2 data set number for each Omnidex master *n*	The size of this array depends on the number of Omnidex master sets (Word 1). It must be large enough to contain one 16-bit word for each Omnidex master set, up to a maximum of 48.
6	Identifies the keyword fields for a given group	data set name or number with a group number in Word 9 (Byte 17)	 element 1 data set number 2 number of keyword fields in the group 3 Word 1 item number of a keyword field Word 2 number of set that contains the field *n*	Data set must be an Omnidex master set or Detail Record indexed (DR) detail set. The size of this array depends on the number of keyword fields in the group (Word 2). It must be large enough to contain two 16-bit words for each keyword field.

Table 2 - 2: ODXINFO Mode values

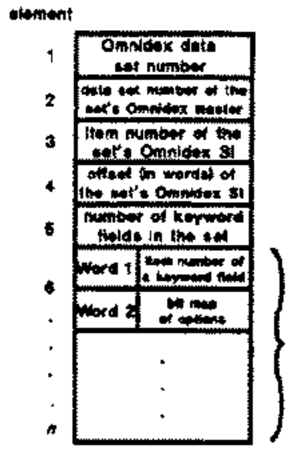
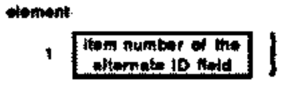
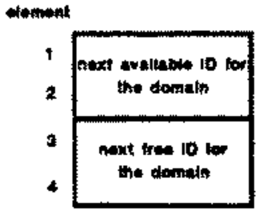
mode	PURPOSE	*qualifier*	*Info* ARRAY CONTENTS	COMMENTS
7	Identifies keyword field options for the specified data set	Omnidex data set name (UPPER CASE) or number	 element 1 Omnidex data set number 2 data set number of the set's Omnidex master 3 item number of the set's Omnidex SI 4 offset (in words) of the set's Omnidex SI 5 number of keyword fields in the set 6 Word 1 item number of a keyword field Word 2 bit map of options n	The size of this array depends on the number of keyword fields for the set (Word 5). It must be large enough to contain two 16-bit words for each keyword field. Bit mapping for Word 2 is as follows: Bit Option 0:1 No Exclude 1:1 No Parse 2:1 Record Specific 3:1 No Translate 4:1 Soundex 5:1 No Deferred 6:1 Batch Indexed 7:1 Type Casting 8:3 data type: 0 = ASCII (U,X or Z) 1 = Integer (I or J) 2 = Packed (P) 3 = Logical (K) 4 = Real (R) 11:5 (not used)
8	Indicates item number for the alternate ID field of an Omnidex master set	Omnidex master set or DR detail set name (UPPER CASE) or number		Contains -1 if the set uses a transparent ID; 2 if the set is a DR detail set.
9	Indicates the next available ID and next free ID values for an Omnidex master set	Omnidex master set name (UPPER CASE) or number	 element 1 2 next available ID for the domain 3 4 next free ID for the domain	

Table 2 - 2: ODXINFO Mode values

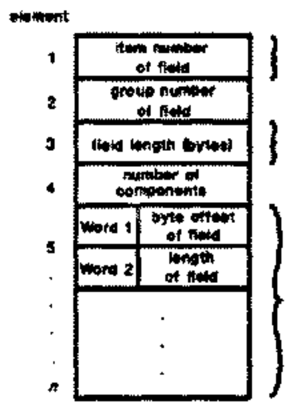
mode	PURPOSE	*qualifier*	*Info* ARRAY CONTENTS	COMMENTS
312	Identifies keyword field options for the specified data set	Omnidex data set name (UPPER CASE) or number, followed by an Omnidex field name or number starting in Word 9 (Byte 17)	 element 1 item number of field 2 group number of field 3 field length (bytes) 4 number of components 5 Word 1 byte offset of field Word 2 length of field *n*	≥1023 for composite fields. Total length for composite field: subitem length for compound field (e.g., 40 for 2X40). The size of this array depends on the number of components (Word 5). It must be large enough to contain two 16-bit words for each component.

Table 2 - 2: ODXINFO Mode values

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested. Word 2 contains the number of words of information returned to the INFO area.

If the call executed successfully, *Status* word 1 is zero (**0**). If *Status* word 1 is **+888**, *Status* word 11 contains the Omnidex condition word. Call DBIERROR or DBIEXPLAIN to interpret the error. See the **Omnidex Condition Word Values** section below for a list of error numbers and their corresponding error conditions.

Info Is an array in which information about Omnidex is returned. The format of this information depends on the *Mode*. Refer back to the mode descriptions to see the contents of the *Info* parameter.

Note: A data set is called an Omnidex data set if it was specified during the Omnidex portion of installation on the data base.

Omnidex Condition Word Values

** CALLING ERRORS **

-500	Illegal mode.
-501	Not an Omnidex data set.
-502	Not an Omnidex master.
-503	Undefined Omnidex group.
-504	No alternate ID field defined.
-505	Not an Omnidex keyword field.
-514	Not an Omnidex database.
-515	Not an IMSAM database.

Please see the section on error messages at the end of this chapter for more information about these conditions.

ODXPRINT

`ODXPRINT (Filename, Keywords, Control, Status, Plabels)`

ODXPRINT prints the contents of an ASCII, Editor, HP Word, Qedit or TDP text file to the line printer. The formal file designator, **ODXPRINT**, is opened with shared lock access.

Note: The HP Word intrinsics must be available in the system library in order to print HP Word documents.

Parameters

Filename Is an array containing the name of the file to be printed. The file name can contain a lockword, group name and account name. It must be terminated with a blank or a semicolon (;).

Keywords The *Keywords* parameter is ignored by ODXPRINT.

Control Is a 10 16-bit-word (20 byte) array. Words 1 - 3 and 8 - 10 are ignored. The remaining words contain the following:

Word	Contents of *Control* parameter
1	A value of 1 can be used to suppress page headings.
2	The number of lines per page.
3	Not used.
4-5	Line number in the file where the printing should start. Files are assumed to have line numbers starting at 1, in increments of 1, whether they are numbered or not. For example, the 20th record in the file is treated as line 20, even if a different editor line number is assigned. A line number of zero (0) means the first line of the file, which is the standard.
6-7	Line number in the file where printing should stop. This is typically set to zero (0). (Zero means the last line in the file.)
8-10	Not used.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (0). If the call failed, *Status* word 1 is +888 and *Status* word 11 Contains 801 or 802.

Plabels	Is the name of a 10 16-bit-word (20-byte) array that must be initialized to all zeros before the first call to ODXPRINT. After this call, the contents of the array must be preserved (not changed) for subsequent calls to ODXPRINT to function properly.

Omnidex Condition Word Values

** CALLING ERRORS **

If the call executed successfully, *Status* word 1 is zero (**0**). If the call failed, *Status* word 1 is **+888** and *Status* word 11 Contains **801** or **802**, as follows:

801	Unable to load the HP Word intrinsics.
802	Unable to open the specified file.
803	Unable to write to the specified file.
804	Unable to read the specified file.
805	Unable to access a session temporary file.

ODXTRANSFER

`ODXTRANSFER ( Base, Mode, Status, Filename, Options )`

ODXTRANSFER copies all the search items (SIs) or IDs qualified by an ODXFIND to a file.

Parameters

Base Is the name of the array used as the *Base* parameter when opening the data base. The first element of the array must contain the base ID returned by DBIOPEN. (See DBIOPEN for more information about the base ID.)

Mode Is a 16-bit-word integer, either **1** or **2** for SIs, or **201** for IDs, as follows:

1 Normal write mode for Omnidex SIs. Overwrites the contents of an existing file.

2 APPEND mode for Omnidex IDs. Appends the SIs to the end of an existing file. If the SIs are transferred to a new file, this parameter is ignored.

<u>Mode Options</u>

In addition, there are several mode options. They are specified by adding the mode option value to the base mode values **1** or **2** only.

For example, ODXTRANSFER Mode 11 performs a normal write of record numbers after an ODXFIND on an RS field in a detail set.

10 Transfers the record number instead of the search item. (ODXFIND on RS fields only.)

20 Transfers the search item and the record number concatenated together. (ODXFIND on RS fields only.)

100 Transfers the data contained in the field specified in the *Options* parameter.

200 External file mode. Transfers IDs, in their internal format, to an external file. This file can then be specified in an ODXFIND keyword list by using the file name, preceded by **$** (**$***filename*), in place of a keyword.

Note that Omnidex IDs cannot be transferred to a file previously created as an SI file because they do not have the same format and file code. Also, APPEND access is not permitted with Omnidex ID files.

☞ Note: Mode Options 10 and 20 are used for record-specific (RS) retrievals only, and can be added to modes 1 and 2 only.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

If the call executed successfully, *Status* word 1 is zero (**0**) and *Status* words 12-13 contain the number of SIs transferred.

If *Status* word 1 is **+888**, *Status* word 11 contains the Omnidex condition word and words 12-13 are set to zero (**0**).

Filename Is the name of an array that contains the name of the file to which the SIs or IDs are to be transferred.

Options Is the name of a 16 16-bit-word (32-byte) array used only with Mode Option 100. The first 16 bytes contain the name of the specified Omnidex master or DR detail data set (terminated by a blank or semicolon). This may be a detail set in two cases:

- ❑ If the detail set was installed with Detail Record (DR) Indexing.
- ❑ If the last ODXFIND was performed on a Record Specific (RS) field. Qualified IDs must still be in RS form, not SI-domain form, if a split retrieval was performed. See **Considerations for Using RS** in the **Data Base Design** chapter of the **OMNIDEX Administrator's Guide** for more information about searches on Record Specific fields.

The second 16 bytes of the *Options* parameter contain the name of the specified field (terminated by a blank or semicolon) to be transferred from each qualified record in the data set.

Discussion

ODXTRANSFER Mode 1 or 2 is used to transfer Omnidex SIs qualified by ODXFIND to a file, which can be used to drive a report writer, or can be used in a Multifind operation.

The file specified by *Filename* is created automatically by ODXTRANSFER. The file is created larger than is required by the current number of qualifying SIs to accept more SIs to be appended later. Still, no disk space is wasted, because unallocated file extents are used for the additional pad space.

You should not use an existing file, because ODXTRANSFER uses MR/NOBUF writes; therefore the file must be created with the correct record size, capacity and blocking factor.

Because ODXTRANSFER uses MR/NOBUF writes for the transfer, and Omnidex SIs are retrieved from the Omnidex indexes, it is very fast.

You can use a file equation to equate the formal file designator for a different file capacity. Yet, you can never use a file equation to specify APPEND access. If you do, an error is returned. Specify Mode 2 for APPEND access.

ODXTRANSFER Mode 101 is used to transfer the contents of a field of an Omnidex master or detail for records qualified by ODXFIND to a file for later use. The field name and data set name are passed through the *Options* parameter. Typically, the field is an SI field in a Detail Record indexed (DR) detail set. The files created by Mode Option 100 are typically used in a Multifind operation.

Because ODXTRANSFER must retrieve each record from the Omnidex master or DR detail, ODXTRANSFER mode 101 or 102 will be much slower than other ODXTRANSFER operations.

ODXTRANSFER Mode 201 is used to transfer Omnidex IDs qualified by ODXFIND to a file for later use. This file of IDs can be reinstated by entering *$filename* in the keyword list for ODXFIND.

For example, if you did a lengthy or complicated keyword retrieval that you knew you would need to do again shortly, you could perform a Mode 201 ODXTRANSFER to save the IDs to a file. When you later needed to do that same retrieval, you would call ODXFIND with *$filename* as the keyword list to reinstate that list of IDs.

You also can use ODXTRANSFER Mode 201 to create files for retrievals across data bases, provided there is an exact record-to-record correspondence between the IDs of the sets involved.

For example, you might store information about customers in two separate data bases. If they have a J2 search item as the Omnidex ID and the ID/SIs correspond exactly, you could do retrievals across the data bases.

If you qualified ID 1, 5 and 10 for the CUSTOMERS set in the first data base, you could use an ODXTRANSFER Mode 201 to save the IDs to a file. You could then restore those IDs in the CUSTOMERS set in the second data base and further qualify by other criteria. There must be a perfect record-to-record relationship between two sets for the second retrieval to be meaningful.

Omnidex Condition Word Values

** CALLING ERRORS **

-800	Bad mode.
-801	File record size must = Omnidex SI size.
-802	File capacity is too small.
-803	Not an ODXTRANSFER ID file.
-804	Insufficient stack space.
-805	Can't transfer IDs to an SI file.
-806	Bad data item reference.
-807	Bad data set reference.
-808	Must be manual master or detail.
-809	Detail chain processing not valid.
-810	Last ODXFIND done on a DR domain.
-811	Unable to transfer field specified.

Please see the section on error messages at the end of this chapter for more information about these conditions.

ODXVIEW

`ODXVIEW  (`*`Filename, Keywords, Control, Status, Plabels`*`)`

ODXVIEW displays the contents of an ASCII, Editor, HP Word, Qedit or TDP text file on the terminal screen.

Note: The HP Word intrinsics must be available in the system library in order to display HP Word documents.

Parameters

Filename	Is an array containing the name of the file to be displayed on the screen. The file name can contain a lockword, group name and account name. It must be terminated with a blank or a semicolon (;).
Keywords	Is an array containing a list of keywords to be highlighted when displayed on the screen. The keywords should be separated by the Omnidex Boolean operators and terminated by a semicolon (;) or blank . A list of keywords for the ODXFIND *Keywords* parameter is in the correct format. See the section on **ODXFIND** in this chapter for more information.

Control Is a 10 16-bit-word (20 byte) array that contains the following:

Word	Contents of *Control* parameter
1	Highlight flag: controls the highlighting of words from the *Keywords* array. 0 = no highlighting. 1 = HP terminal highlighting. 3 = ANSI terminal highlighting.
2	Number of lines displayed on the first screen page. For Versions 2.03 and 2.04, this was 24 lines per screen page. Versions 2.05 and 2.06 have been enhanced to display 100 lines per screen page, to use the advanced display capabilities of newer terminals.
3	Number of lines displayed on subsequent screen pages. For Versions 2.03 and 2.04, this was 24 lines per screen page. Versions 2.05 and 2.06 have been enhanced to display 100 lines per screen page, to use the advanced display capabilities of newer terminals.
4-5	Line number in the file where the display should start. Files are assumed to have line numbers starting at 1, with increments of 1, whether they are numbered or not. For example, the 20th record in the file is treated as line 20, even if a different editor line number is assigned. A line number of zero (0) means the first line of the file. A negative (i.e., -1) line number value means scan the file starting at the specified line (here, 1) until a word in the *Keywords* array is found. Then back up half a page, and begin displaying the document.
6-7	Line number in the file where display should stop (zero (0) means the last line in the file).
8-10	Not used.

Status Is the name of the array used as the *Status* parameter in the IMSAM or Omnidex procedure call about which information is requested.

Plabels Is the name of a 10 16-bit-word (20-byte) global array that must be initialized to all zeros before the first call to ODXVIEW. After this call, the contents of the array must be preserved (not changed) for subsequent calls to ODXVIEW to function properly.

Word 8 of *Plabels* is used to specify an inactivity time-out in seconds. Values can range from 60 to 600 (seconds), or 0 (zero) which disables the time-out.

Discussion

ODXVIEW displays one screen page of text, then pauses until a key is pressed at the terminal. The number of lines in a screen page is determined by words 2 and 3 of the *Control* parameter. The file is opened with shared lock access.

One-letter commands may be entered from the terminal during the execution of a program that has called ODXVIEW. They are:

B -	Rewind to the beginning of file. Resets the F command.
E -	Exit ODXVIEW.
F -	Go back to the first screen page.
L -	Go to the last screen page.
N -	Display next screen page.
P -	Display previous screen page.
R -	Rewind to the first screen page.
S -	Scan forward to next keyword occurrence.
1-9 -	Display the next 1 to 9 lines.
↵ -	Display the next screen page.

Any other character causes ODXVIEW to close the file and return to the calling program. After the last line of a file is displayed, ODXVIEW waits for the user to enter the next command.

Omnidex Condition Word Values

** CALLING ERRORS **

If the call executed successfully, *Status* word 1 is zero (**0**). If the call failed, *Status* word 1 is **+888** and *Status* word 11 Contains **801** or **802**, as follows:

801	Unable to load the HP Word intrinsics.
802	Unable to open the specified file.
803	Unable to write to the specified file.
804	Unable to read the specified file.
805	Unable to access a session temporary file.

Error Messages

Introduction

This section lists the calling errors and exceptional conditions for IMSAM and Omnidex. It includes the error message number, the type of call (IMSAM or Omnidex) when the error or exceptional condition occurred, the text of the error message, an explanation of the error and an action to take to correct the error.

In application programs that call IMSAM or Omnidex intrinsics, you can look at IMAGE *Status* word 1 to see if an error has occurred. If *Status* word 1 is +999, an IMSAM error has occurred. If it is **+888**, an Omnidex error has occurred. DBIEXPLAIN or DBIERROR should be called to interpret the error. The error code is contained in word 11 of the *Status* array.

Conventions

Error message numbers indicate which intrinsic was being processed when the error or exceptional condition occurred. All error messages consist of 3 digits in the range 100 through 999, positive or negative. The numbers are assigned to the intrinsics as follows:

100s	DBIPUT messages
200s	DBIFIND or ODXFIND messages
300s	DBIGET, ODXGET or ODXGETWORD messages
400s	DBIDELETE messages
500s	DBIINFO, DBILOCK or ODXINFO messages
600s	DBIUPDATE messages
700s	Miscellaneous messages
800s	ODXPRINT, ODXTRANSFER and ODXVIEW messages
900s	DBIOPEN messages

A positive number indicates an exceptional condition. For example, message number 310 is issued by DBIGET (indicated by the 3 in the first digit) and indicates that the beginning of file has been reached on a descending sequential read.

A negative number indicates a calling error - an error encountered in the syntax of the call to the intrinsic. For example, message number -100 is issued by DBIPUT (indicated by the 1 in the first digit) and indicates that an invalid mode was specified.

Note that error numbers may refer to messages for IMSAM or Omnidex intrinsics, or both, depending on the type of data set. Error numbers that have a message for an IMSAM and an Omnidex intrinsic are listed twice, separated by the word "OR".

Also note that some errors and exceptional conditions are repeated for each intrinsic. For example, the error numbers 196, 296, 396, 496 and 696 all indicate that the ILCB or OLCB is damaged.

Note: Errors that do not appear in this list are internal programming errors. DISC phone-in consulting should be called for technical support.

Configuring Error Messages

Most of the error messages for calls to IMSAM and Omnidex intrinsics are contained in a file called ODXERROR in the PUB.DISC account. Each line of the file corresponds to an error message. The first character of each line, I or O, indicates whether the error message is an IMSAM or Omnidex error. Next the error number is listed, followed by the error message. For example:

```
I 130Number of tree levels exceeds maximum
I 141IMSAM tree is full
I-100Illegal Mode specified
I-101An @ itemlist is required in discrete mode
I 210Beginning of file
```

The errors are listed in order by error number; the IMSAM errors are listed first, followed by the Omnidex errors.

You can modify many of the error messages if you want. To do so, load the file into an EDITOR compatible text editor, and change the text that follows each number. The only restrictions that apply to changing error messages are:

- ❑ **Some errors, most notably those with numbers ending in 96 and 97, are embedded in code and therefore not configurable.** Errors that cannot be reconfigured are so noted in this chapter by an **(nc)** appended to their numbers.
- ❑ **Your error message should not exceed one line.**
- ❑ **Do not modify the error numbers as it causes erroneous error messages to be returned.**

Errors Listed by Number

100s - DBIPUT

`130`	`IMSAM`	`Number of tree levels exceeds maximum`

The maximum number of levels in an IMSAM B-tree structure is 5. The last key added has caused a level to split, and the B-tree requires a sixth level, which exceeds the maximum.

ACTION: You can increase the usage of each level by repopulating the tree with the DBIUTIL INDEX;LOAD=*nn*% command and specifying a larger load factor. This may decrease the number of levels required for the B-tree. See the DBIUTIL section of the **Utilities** chapter of the **OMNIDEX Administrator's Guide** for more information about INDEX operations.

You also could deinstall OMNIDEX, and reinstall it with a larger tree block size. See the **Performance Recommendations** and **Reinstallation** sections in the **Installation** chapter of the **OMNIDEX Administrator's Guide** for more information about tree block size.

`141`	`IMSAM`	`IMSAM tree is full`

The number of entries in an IMSAM B-tree index set has reached the capacity for that set. Therefore no more records can be added to the IMAGE data set that contains the IMSAM key.

ACTION: You must increase the data set capacity of the IMSAM B-tree set. Use the CAP command of DBMGR or a similar utility, or change the capacity in the data base schema then recreate the data base using a DBUNLOAD followed by a DBLOAD.

`196 (ac)`	`IMSAM or Omnidex`	`ILCB is damaged or OLCB is damaged`

During processing, the IMSAM local control block (ILCB) or the Omnidex local control block (OLCB) in the stack has been overlaid.

ACTION: Call the DISC response center.

```
197 (nc)        IMSAM or          Bad base ID, or data base not
                Omnidex           opened using DBIOPEN
```

This problem usually occurs after a program calls an IMAGE DBOPEN instead of a DBIOPEN or if no open was called. DBIOPEN initializes the IMSAM and Omnidex local control blocks (ILCB and OLCB), which are required for processing.

ACTION: Make sure you use a DBIOPEN call in your program or use DBOPEN and run the program using the call conversion library (for example, enter :RUN program.CC.DISC;LIB=G).

```
-100            IMSAM             Illegal mode specified
```

An illegal mode was specified in the *Mode* parameter of a call to DBIPUT.

ACTION: See the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for valid DBIPUT mode values.

```
-101            IMSAM             An @ itemlist is required in
                                  discrete mode
```

DBIPUT requires an at-sign (@) item list for a discrete mode (Mode 201) Put, which indexes all the key fields in the data set without adding the entry itself to the set.

ACTION: An at-sign (@) itemlist must be used when calling DBIPUT in discrete mode. Also, the *Buffer* parameter must contain a full entry with all fields in set definition order. See the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for more information.

```
-111            Omnidex           Omnidex ID cannot exceed next
                                  available ID
```

A DBIPUT was issued to add a record with an Omnidex ID greater than the next-available Omnidex ID. The next-available Omnidex ID is maintained by Omnidex.

ACTION: You can let Omnidex initialize the Omnidex ID value by placing the value zero in the ID field before calling DBIPUT. Omnidex automatically increments the next-available ID. Or you can use the ODXUTIL NEXT command to increase the next-available Omnidex ID to a value above the one you are trying to add.

```
-112          Omnidex          An @ itemlist or the
                               equivalent must be used
```

DBIPUT requires an at-sign (@) item list or an equivalent explicit list for Puts to an Omnidex data set.

ACTION: Use an at-sign (@) list when calling DBIPUT. Or, use an explicit list containing the data set's items in the order they appear in the set.

```
-113          Omnidex          Omnidex ID cannot exceed
                               8,388,607
```

DBIPUT was called with a value greater than 8,388,607 in the Omnidex ID key field.

ACTION: You must add the record using a lower value in the Omnidex ID field or make sure your program is not using ASCII characters for the Omnidex IDs. ASCII characters in a double- word integer field yield a value greater than 300,000,000.

```
-114          Omnidex          Omnidex ID must be greater
                               than 0
```

A discrete mode DBIPUT was called for a record that contained a value of 0 or less in the Omnidex ID field.

ACTION: Check the data and make sure all the Omnidex ID values are greater than 0.

```
-115          Omnidex          ID cross reference set is
                               full
```

An XODX# ID/SI cross reference set is full, so you cannot add another record until its capacity is increased.

ACTION: You must increase the capacity of this XODX# set.

```
-116          Omnidex          ID cross reference set is
                               corrupt
```

An XODX# ID/SI cross reference set has been corrupted.

ACTION: You must run ODXUTIL and perform an INDEX or BF operation on the master with the corrupt XODX# set.

-117	Omnidex	Maximum chain length for details with RS indexing is 255

An attempt was made to add the 256th record in a chain to a detail set with Record Specific keyword fields. This is not allowed.

ACTION: You can archive and delete other records in the chain and then add the new record.

-119	Omnidex	Deferred updates restricted to one domain

An attempt was made to add records to more than one Omnidex domain while the data base was open with Mode Option 100 for deferred update. Only one Omnidex domain (master set and related detail sets) can be updated at a time in deferred update mode.

ACTION: Complete the deferred update on one domain before adding records to another domain.

200s - DBIFIND or ODXFIND

201	Omnidex	CTRL-Y interrupt

A CTRL-Y interrupt occurred during an ODXFIND. ODXFIND returns to the calling program with the qualifying count set to 0 (zero) and the current list of qualifying IDs unchanged.

ACTION: No action is required. You can do another retrieval.

210	IMSAM	Beginning of file

During descending order sequential retrieval, the first entry has been retrieved and there are no more records to read.

ACTION: No action is required. You can do another retrieval.

211	IMSAM	End of file

During ascending order sequential retrieval, the last entry has been retrieved and there are no more records to read.

ACTION: No action is required. You can do another retrieval.

212	Omnidex	**Stack overflow: Control buffers**

There was not enough stack space for the minimum number of internal buffers required by ODXFIND.

ACTION: Increase the MAXDATA, if possible, or reduce the amount of stack used by your program for data storage. For more information, see the **Stack Size** section of the **General Considerations** chapter of this manual.

213	**IMSAM**	**IMSAM tree is empty**

The IMSAM B-tree for partial-key and sequential access is empty, therefore IMSAM cannot retrieve records.

ACTION: To repopulate the IMSAM tree, use the DBIUTIL INDEX command. See the DBIUTIL part of the **Utilities** chapter of the **OMNIDEX Administrator's Guide** for more information.

213	Omnidex	Too many keywords in list

There are too many keywords in the *Keywords* parameter for the ODXFIND to be executed in stack. Approximately 1000 words of stack are required for each keyword. Therefore the maximum number of keywords that can be used in the keyword list for an ODXFIND is from 20 to 30.

ACTION: First check the MAXDATA and make sure it is around 31,232.

If the MAXDATA is a high number, and you still get error 213, you must remove some keywords from the list. If necessary, break the retrieval into two or more parts and use the SAMELIST operator to continue the retrieval.

217 (nc)	**IMSAM**	**Key not found**

The value supplied to IMSAM for retrieval had no matching entries in the data base. If this was a partial-key value, there were no entries with that value as the first part of their key.

ACTION: No action is required. You can do another retrieval.

```
217            Omnidex          No entries contain KEYWORD
```

KEYWORD is a keyword specified in an ODXFIND that does not exist in the specified field for any entry in this set. The failing keyword is passed back in *Status* array words 16 - 21 and is included in the error message returned by DBIERROR.

ACTION: If you know that that keyword exists in a record, check the keyword list and make sure it is terminated by a semicolon or a blank. If the keyword does not exist, no action is required and you can do another retrieval.

```
296 (nc)       IMSAM or         ILCB is damaged   or   OLCB
               Omnidex          is damaged
```

During processing, the IMSAM local control block (ILCB) or the Omnidex local control block (OLCB) in the stack was overlaid.

ACTION: Call the DISC response center.

```
297 (nc)       IMSAM or         Bad base ID, or data base not
               Omnidex          opened using DBIOPEN
```

This problem usually occurs after a program calls an IMAGE DBOPEN instead of a DBIOPEN or if no open was called.

DBIOPEN initializes the IMSAM and Omnidex local control blocks (ILCB and OLCB), which are required for processing.

ACTION: Make sure you use a DBIOPEN call in your program or use DBOPEN and run the program using the call conversion library (for example, enter :RUN program.CC.DISC;LIB=G).

```
-200           IMSAM            Illegal mode specified
```

An illegal mode was specified in the *Mode* parameter of a call to DBIFIND.

ACTION: See the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for valid DBIFIND mode values.

```
-201           IMSAM            Data set not an IMSAM Detail
```

In a call to DBIFIND that specified an IMSAM retrieval mode, the data set you specified is either not a detail data set or has no IMSAM key fields.

ACTION: Check the *Mode* and *Dset* parameters you specified in the call to DBIFIND. On a non-IMSAM detail data set, a Mode 1 DBIFIND is the only valid mode. To use IMSAM retrieval modes in DBIFIND, you must specify a detail data set that has one or more IMSAM keys.

```
-201        Omnidex        KEYWORD is an excluded word
```

KEYWORD is a word contained in the keyword list of a call to ODXFIND. It is an excluded word and therefore not a valid qualifier.

ACTION: Eliminate the word from the keyword list. If you want the word to be indexed, you need to delete it from your excluded word list. (See the ODXUTIL XCLUDE command in the **Utilities** chapter of the **OMNIDEX Administrator's Guide** for more information.) Also use the ODXUTIL INDEX command to reindex the data base so that existing entries that contain the word are indexed by that word.

```
-202        IMSAM          Key value exceeds defined key
                           length
```

In the *Mode* parameter of a call to DBIFIND, you specified a key value whose length exceeded the defined key length. You can specify a key value from one byte up to the full defined length of the key, but you cannot exceed that length. For example, Mode 310 (or -320) specifies a key value of 10 words (20 bytes), which is valid only for a key length of 20 bytes or more.

ACTION: Check the defined key length for the data set in question and modify the mode value accordingly.

```
-202        Omnidex        Illegal ODXFIND mode
```

An illegal mode was specified in the *Mode* parameter of a call to ODXFIND.

ACTION: See the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for valid ODXFIND mode values.

```
-203        Omnidex        Data set is not an Omnidex set
```

The data set specified in a call to ODXFIND is not an Omnidex set.

ACTION: Check the data set name specified.

```
-204        IMSAM          Item is not an IMSAM key
```

An IMSAM mode was specified in a call to DBIFIND, but the item name specified in the *Item* parameter is not defined as an IMSAM key field.

ACTION: Check the value of the *Mode* parameter and the *Item* parameter. Either change the *Mode* parameter to a value of 1 (non-IMSAM Find) or change the value of Item to an IMSAM key, whichever is appropriate.

```
-204          Omnidex          Illegal use of NOT operator
```

The NOT operator (-) was inappropriately specified in a keyword list. The NOT operator cannot be specified with the first keyword in a list unless preceded by the samelist operator (*), nor can it be specified after an OR operator (+).

ACTION: Correct the keyword list.

```
-205          Omnidex          Bad keyword list
```

ODXFIND was called using a syntactically incorrect keyword list.

ACTION: Examine the keyword list for errors. If none are readily apparent, review ODXFIND in the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide.** A common error is an extra operator, like the double commas in **"THIS,,THAT"**.

```
-207          Omnidex          Keywords must start with an
                               alphanumeric character or
                               decimal point
```

The first character of a keyword list passed to ODXFIND was invalid. It must be an alphanumeric character (letters or numbers) or a decimal point.

ACTION: Correct the list.

```
-208          Omnidex          Start and stop values must
                               have the same number of digits
```

A keyword list specified a range of two numeric values, but the two values had from different number of digits. This error applies only to ASCII fields or groups containing ASCII fields.

ACTION: Correct the values in the range so that they have the same number of digits. Or, if you are specifying a partial keyword, make sure the partial value is followed by an at-sign (@). For example, **8000:9000**, **8000:9@** and **8000:8599.63** are allowed, but **8000:90000** is not.

```
-209          Omnidex          Bad item name specified
```

An invalid item name or number was specified as the keyword field in a call to ODXFIND.

ACTION: Correct the *Field* parameter.

-210	`Omnidex`	`Item is not an Omnidex keyword field`

An item name specified in an ODXFIND call is not defined as an Omnidex keyword field in the data set.

ACTION: Determine what the correct item name is and correct the *Item* parameter.

-211	`Omnidex`	`Only one generic keyword or one keyword range permitted`

This message appears only when a Mode 10 or 11 ODXFIND is performed. These are discrete modes, since they return information about the keywords only and not about the qualifying Omnidex IDs. Because these modes are not used for subsequent data record retrieval, only one generic keyword or range is appropriate.

ACTION: Correct the value in the keyword list. Or, if the *Mode* parameter is inappropriate, change the mode to 1.

-212	`IMSAM`	`No current key`

A DBIFIND Mode 92 was called to locate the current chain head, but there was no established current entry.

ACTION: Make sure that a previous DBIFIND call was issued to establish the chain. Also, the current chain head may have been deleted between DBIFIND calls. If this is the case, you can reissue a DBIFIND locating the next chain head using a Mode 90 or 91 DBIFIND. For more details on these modes, see the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide.**

-212	`Omnidex`	`List must contain a generic keyword or keyword range`

This message appears only when a Mode 10 or 11 ODXFIND is performed. These are discrete modes, since they return information about the keyword only, and not about the qualifying Omnidex ID. Because these modes are not used for subsequent data record retrieval, it is assumed that information about more than one keyword is desired.

ACTION: Correct the value in the keyword list. Or, if the *Mode* parameter is inappropriate, change the mode to 1 or 2.

```
-213          Omnidex          Samelist not allowed in
                               discrete modes (10 and 11)
```

This message appears only when a Mode 10 or 11 ODXFIND is performed. These modes do not support the operator, since they are discrete modes and return information about the keywords only, not about the qualifying Omnidex IDs.

ACTION: Correct the keyword list or change the mode for the ODXFIND.

```
-214          Omnidex          Not an Omnidex data base
```

ODXFIND was called with a *Base* parameter that is not an Omnidex data base.

ACTION: Install Omnidex on the data base or change the *Base* parameter to an Omnidex data base.

```
-216          Omnidex          Generic keywords not allowed
                               on binary fields
```

Binary fields that are keyworded for Omnidex cannot be retrieved using a generic keyword. Only individual keyword values or ranges can be used.

ACTION: Change the keyword to a keyword, Boolean combination or keyword range.

```
-217          Omnidex          Keyword contains a
                               non-numeric character
```

A keyword for a binary field contains an ASCII character.

ACTION: Correct the keyword so that it is all numeric.

```
-218          Omnidex          Missing keyword in the list
```

A keyword is missing in the keyword list. There is an extraneous comma or two commas together.

ACTION: Correct the keyword list.

```
-219          Omnidex          Unmatched quote
```

A keyword or keyword range on a No Parse field that includes blanks is missing a quotation mark.

ACTION: Re-enter the keyword with the appropriate quotation marks.

-220	`Omnidex`	`Missing terminator (semicolon or blank)`

The keyword list must end with a semicolon (;) or blank ( ).

ACTION: Re-enter the keyword with the appropriate terminator.

-221	`Omnidex`	`Start and stop values must have the same sign`

The start and stop values for a range must be both negative or both positive for a binary field.

ACTION: Correct the range expression.

-222	`Omnidex`	`Split retrievals not allowed with Samelist OR`

You cannot do a Samelist OR retrieval on a Record Specific field after a standard Omnidex record-complex retrieval, or vice versa.

ACTION: Use a Samelist AND or NOT, if appropriate. Or try to limit the retrieval to only Record Specific or SI-domain fields.

-223	`Omnidex`	`Dset must be the same as in previous RS retrieval`

An attempt was made to do a Record Specific retrieval on a different detail data set from a previous RS retrieval.

Successive RS retrievals must be done on the same detail set because the qualifying count is the number of records in a given set that meet the retrieval criteria on that set.

ACTION: Change the *Dset* parameter in the current ODXFIND call to the same data set as the previous RS ODXFIND call or do an ODXFIND Mode 30 compaction between the RS ODXFINDs.

```
-224            Omnidex          Not a Soundex field
```

An attempt was made to do a Soundex retrieval on a field that was not installed with Soundex.

If the retrieval was done on a keyword group, the Soundex field was not specified as the item for retrieval.

ACTION: Do not enter a question mark (?) at the end of the keyword during a retrieval, or install Soundex on the field. If retrieval was done in DATADEX on a keyword group, reconfigure the prompt for that group (via DATADEX's CONFIG command), specifying the Soundex field as the first item in the group for retrieval. If the retrieval was done through a call to ODXFIND, specify the Soundex field in the *Field* parameter.

```
-231            Omnidex          Not an ODXTRANSFER ID file
```

An attempt was made to use the External File operator ($) on a file other than an ODXTRANSFER ID file.

ACTION: Recreate the file by first requalifying the appropriate keywords. In DATADEX, use the XFER OMNIDEX (XO) command with the ;ID option. When prompted, specify a new file to receive the record numbers. DATADEX automatically sizes the file. Application programs that call ODXTRANSFER should pass Mode 201 to create a file suitable for retrieval using the "$" operator.

300s - DBIGET, ODXGET or ODXGETWORD

```
301             IMSAM            Critical flag is set
```

An IMSAM search was conducted while the B-trees were in a critical update state.

ACTION: Try again later.

```
310             IMSAM            Beginning of file
```

During descending order sequential retrieval, the first entry has been retrieved and there are no more records to read.

ACTION: No action is required. You can do another retrieval.

```
310           Omnidex          Beginning of ID list
```

During descending order retrieval of the Omnidex IDs, the first ID in the list has been retrieved and there are no more IDs to retrieve.

ACTION: No action is required. You can do another retrieval.

```
311           IMSAM            End of file
```

During ascending order retrieval, the last entry has been retrieved and there are no more records to read.

ACTION: No action is required. You can do another retrieval.

```
311           Omnidex          End of ID list
```

During ascending order sequential retrieval of the Omnidex IDs, the last ID has been retrieved and there are no more IDs to retrieve.

ACTION: No action is required. You can do another retrieval.

```
313           IMSAM            IMSAM tree is empty
```

The IMSAM tree for partial-key and sorted-sequential access is empty, therefore, IMSAM can not retrieve records.

ACTION: To repopulate the IMSAM tree, use the DBIUTIL INDEX command. See the DBIUTIL part of the **Utilities** chapter of the **OMNIDEX Administrator's Guide** for more information about BUILD.

```
314           Omnidex          End of range
```

An ODXGETWORD listing of keywords has reached the last keyword in the range.

ACTION: No action is required. You can do another retrieval.

```
317           IMSAM            Key not found
```

The value supplied to IMSAM for retrieval has no matching entries in the data base. If this was a partial key value, there are no entries with that value as the first part of their key.

ACTION: No action is required. You can do another retrieval.

`396 (nc)`	`IMSAM or Omnidex`	`ILCB is damaged or OLCB is damaged`

During processing, the IMSAM local control block (ILCB) or the Omnidex local control block (OLCB) in the stack has been overlaid.

ACTION: Call the DISC response center.

`397 (nc)`	`IMSAM or Omnidex`	`Bad base ID, or data base not opened using DBIOPEN`

This problem usually occurs after a program calls an IMAGE DBOPEN instead of a DBIOPEN or if no open was called.

DBIOPEN initializes the IMSAM and Omnidex local control blocks (ILCB and OLCB), which are required for processing.

ACTION: Make sure you use a DBIOPEN call in your program or use DBOPEN and run the program using the call conversion library (for example, enter :RUN program.CC.DISC;LIB=G).

`-300`	`IMSAM or Omnidex`	`Illegal mode specified`

An illegal mode was specified in the *Mode* parameter of a call to DBIGET.

ACTION: See the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for valid DBIGET mode values.

`-301`	`IMSAM`	`Data set not an IMSAM data set`

An IMSAM retrieval mode was specified in a call to DBIGET, but the data set specified in the *Dset* parameter has no IMSAM key fields.

ACTION: Correct the *Dset* parameter of the DBIGET call or change the *Mode* parameter.

`-301`	`Omnidex`	`Target array is too small`

The buffer array address passed to the *Target* parameter of ODXGET is the address of an area in storage. When the value in the *ID-count* parameter was added to it, a bounds violation occurred.

ACTION: Check the buffer area named in the *Target* parameter. Make sure that you have defined it large enough to accommodate the *ID-count* IDs.

-302	IMSAM	Key value exceeds defined key length

The *Mode* parameter used in a call to DBIGET specified a key value whose length exceeded the defined key length. The mode can specify a key value from one byte up to the full defined length of the key, but may not exceed that length. For example, a Mode 310 (or -320) specifies a key length of 10 words (20 bytes), which is valid for a key that is 20 bytes or longer.

ACTION: Check the defined key length for the data set in question and modify the mode value accordingly.

-302	Omnidex	No preceding Mode 10/11 ODXFIND

ODXGETWORD was called to retrieve a keyword without first calling ODXFIND in discrete mode (Mode 10 or 11).

ACTION: Make sure a discrete mode ODXFIND is called before ODXGETWORD. Note that a normal mode (Mode 1 or 2) call to ODXFIND cancels a previous discrete mode ODXFIND.

-303	Omnidex	No preceding ODXFIND

ODXGET was called to retrieve an Omnidex SI without first calling ODXFIND Mode 1 or 2.

ACTION: Make sure an ODXFIND Mode 1 or 2 is called before ODXGET.

-304	IMSAM	Item is not an IMSAM key

An IMSAM mode was specified in a call to DBIGET, but the item specified as the key (in words 9-16 of the *Dset* parameter) is not defined as an IMSAM key field.

ACTION: Check the values of the *Mode* parameter and the *Dset* parameter. Correct them as appropriate.

-304	Omnidex	Illegal count

The value for the *SI-Count* parameter in an ODXGET is not valid. It must be in the range from 1 to 4096.

ACTION: Change the value for the *SI-Count* parameter to a valid number.

```
-305            IMSAM             Image search items require a
                                  DBIFIND followed by chained
                                  DBIGETs
```

An IMSAM mode DBIGET must not specify an IMAGE search item in a detail, even if that search item is an IMSAM key in an IMSAM master.

ACTION: Perform a DBIFIND on the detail data set and specify the desired search item. If the search item is an IMSAM key, any of the IMSAM retrieval modes can be used. If the search item is not an IMSAM key, only Mode 1 DBIFINDs are allowed.

Then call DBIGET repeatedly using Mode 5 or 6 (forward and backward chained reads, respectively) to retrieve each entry in the chain.

```
-305            Omnidex           Preceding ODXFIND was not
                                  Record Specific
```

An ODXGET Mode Option 10 or 20 attempted to return Record Specific information, but the previous ODXFIND was not performed on a Record Specific field.

ACTION: Use a Mode 1 through 5 ODXGET or make sure the ODXFIND is being performed on a Record Specific field.

```
-312            IMSAM             No current key
```

Either a previous DBIFIND was not called to establish a current key or the current key value has been deleted.

ACTION: Ensure that a DBIFIND was called to establish a current key value.

```
-314            Omnidex           Not an Omnidex data base
```

The *Base* parameter in a call to ODXGET specifies a data base that does not have Omnidex installed.

ACTION: Check the *Base* parameter to make sure it specifies the correct data base. Also check the data base to make sure that Omnidex was installed.

400s - DBIDELETE

`496 (nc)`	`IMSAM or Omnidex`	`ILCB is damaged or OLCB is damaged`

During processing, the IMSAM local control block (ILCB) or the Omnidex local control block (OLCB) in the stack has been overlaid.

ACTION: Call the DISC response center.

`497 (nc)`	`IMSAM or Omnidex`	`Bad base ID, or data base not opened using DBIOPEN`

This problem usually occurs after a program calls an IMAGE DBOPEN instead of a DBIOPEN or if no open was called.

DBIOPEN initializes the IMSAM and Omnidex local control blocks (ILCB and OLCB), which are required for processing.

ACTION: Make sure you use a DBIOPEN call in your program or use DBOPEN and run the program using the call conversion library (for example, enter :RUN program.CC.DISC;LIB=G).

`-400`	`IMSAM`	`Illegal mode specified`

An illegal mode was specified in the *Mode* parameter of a call to DBIDELETE.

ACTION: See the **Intrinsics** of the **OMNIDEX Programmer's Guide** chapter for valid DBIDELETE mode values.

`-402`	`IMSAM`	`No current key`

The current key has already been deleted or the call was not preceded by a normal or discrete mode DBIGET or DBIPUT.

ACTION: Establish a current key with a DBIGET call before issuing the DBIDELETE call.

`-403`	`IMSAM`	`DBIDELETEs of RS indexed details not permitted`

A delete was attempted on a record in a detail set that uses the Record Specific (RS) indexing option. This is not allowed.

ACTION: Use an IMAGE-only delete to delete the record by running Query or another program without call conversion.

Then erase the indexes and repopulate them using ODXUTIL.

500s - DBIINFO, DBILOCK or ODXINFO

```
596 (nc)       IMSAM            ILCB is damaged
```

During processing, the IMSAM local control block (ILCB) in the stack has been overlaid.

ACTION: Call the DISC response center.

```
597 (nc)       IMSAM            Bad base ID, or data base not
                                opened using DBIOPEN
```

This problem usually occurs after a program calls an IMAGE DBOPEN instead of a DBIOPEN or if no open was called.

DBIOPEN initializes the IMSAM and Omnidex local control blocks (ILCB and OLCB), which are required for processing.

ACTION: Make sure you use a DBIOPEN call in your program or use DBOPEN and run the program using the call conversion library (for example, enter :RUN program.CC.DISC;LIB=G).

```
-500 (nc)      Omnidex          Illegal mode
```

An illegal mode was specified in the *Mode* parameter of a call to DBILOCK.

ACTION: See the **Intrinsics** of the **OMNIDEX Programmer's Guide** chapter for valid DBILOCK mode values.

```
-501           IMSAM            Dset is not an IMSAM data set
```

The data set specified in the *Qualifier* parameter for a Mode 311 or 312 call to DBIINFO is not defined as an IMSAM data set.

ACTION: Check the data set specified in the DBIINFO call.

```
-501           Omnidex          Not an Omnidex data set
```

The data set specified in the *Qualifier* parameter for a call to ODXINFO is not defined as an Omnidex data set.

ACTION: Check the data set specified in the ODXINFO call.

```
-502          IMSAM          Item is not an IMSAM key
```

The data set specified in the *Qualifier* parameter for a Mode 312 or 313 call to DBIINFO is not defined as an IMSAM key.

ACTION: Check the key specified in the DBIINFO call.

```
-502          Omnidex        Not an Omnidex master
```

The data set specified in the *Qualifier* parameter for a Mode 3, 4 or 6 call to ODXINFO is not defined as an Omnidex master data set.

ACTION: Check the data set specified in the ODXINFO call.

```
-503          Omnidex        Undefined Omnidex group
```

The group number specified in the *Info* parameter of Mode 6 call to ODXINFO does not exist.

ACTION: Check the *Info* parameter and make sure you specified the correct data set. This must be a master data set name, even if the group resides only in an Omnidex detail set. Also make sure the group number is correct.

```
-504          Omnidex        No alternate ID field defined
```

The master set specified in an ODXINFO Mode 8 does not have alternate field defined.

ACTION: Make sure you specify a master that contains an alternate ID.

```
-505 (nc)     Omnidex        Not an Omnidex keyword field
```

The item specified in an ODXINFO Mode 312 is not an Omnidex keyword field.

ACTION: Make sure you specify a field that has Omnidex installed on it.

```
-505          IMSAM          Target array too small
```

The *Buffer* parameter in DBIINFO, or the *Info* parameter in ODXINFO, is too small to accept the information requested.

ACTION: See the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for information about DBIINFO or ODXINFO

```
-514            Omnidex         Not an Omnidex data base
```

The *Base* parameter in a call to ODXINFO specifies a data base that does not have Omnidex installed.

ACTION: Check the *Base* parameter. If it is correct, make sure that the data base has Omnidex installed.

```
-515            Omnidex         Not an IMSAM data base
```

The *Base* parameter in a call to ODXINFO specifies a data base that does not have Omnidex or IMSAM installed.

ACTION: Check the *Base* parameter to make sure it specifies the correct data base. Also check the data base to make sure that Omnidex or IMSAM was installed.

600s - DBIUPDATE

```
601             IMSAM or        User buffer not large enough
                Omnidex         for the requested data.
```

The current data set's record layout in the IMSAM rootfile does not match the actual record layout any longer. A field has probably been added to the data set without reinstalling Omnidex and IMSAM.

ACTION: Reinstall Omnidex and IMSAM

```
696 (nc)        IMSAM or        ILCB is damaged   or OLCB is
                Omnidex         damaged
```

During processing, the IMSAM local control block (ILCB) or the Omnidex local control block (OLCB) in the stack has been overlaid.

ACTION: Call the DISC response center.

`697 (nc)`	`IMSAM or Omnidex`	`No ILCB exists or No OLCB exists`

This problem usually occurs after a program calls an IMAGE DBOPEN instead of a DBIOPEN or if no open was called.

DBIOPEN initializes the IMSAM and Omnidex local control blocks (ILCB and OLCB), which are required for processing.

ACTION: Make sure you use a DBIOPEN call in your program or use DBOPEN and run the program using the call conversion library (for example, enter :RUN program.CC.DISC;LIB=G).

`-611`	`IMSAM`	`Alternate ID field cannot be modified`

DBIUPDATE cannot modify the field that contains an alternate ID for an Omnidex master.

ACTION: Do not modify an alternate ID field.

700s - MISCELLANEOUS

`-701`	`IMSAM or Omnidex`	`Procedure not callable`

An attempt was made to call an internal intrinsic that is not accessible to the calling program.

ACTION: Do not try to call the internal intrinsics.

`702 (nc)`	`IMSAM or Omnidex`	`Unable to load`

An attempt was made to run a Compatibility Mode program through the switch stubs, but XL.PUB was not found.

ACTION: Copy XL.PUB.DISC to the PUB group of the calling program's account. See **OMNIDEX Intrinsic Switch Stubs** in the **Programming** chapter of this manual.

800s - ODXTRANSFER OR ODXVIEW

801	Omnidex	`Unable to load the HP Word intrinsics`

An attempt was made to view an HP Word file, but the HP Word intrinsics, which are purchased separately from HP, were not found.

ACTION: Make sure the HP Word intrinsics are in XL.PUB.SYS or SL.PUB.SYS.

802	Omnidex	`Unable to open the specified file`

An attempt was made to open a file that cannot be found or that has a lockword.

ACTION: Make sure the file exists and you include any required lockwords in the *Filename* parameter.

803	Omnidex	`Unable to write to the specified file`

An attempt was made to write to the specified file and a file system error occurred.

ACTION: Check displayed file system error and take appropriate.

804	Omnidex	`Unable to read the specified file`

An attempt was made to write to the specified file and a file system error occurred.

ACTION: Check displayed file system error and take appropriate.

805	Omnidex	`Unable to access a session temporary file`

An attempt was made to write to the specified file and a file system error occurred.

ACTION: Check displayed file system error and take appropriate.

-800	Omnidex	`Bad mode specified`

The integer value passed in the *Mode* parameter does not correspond to any recognizable mode value.

ACTION: Check the size, data type and value of the *Mode* parameter.

`-801`	`Omnidex`	`File record size must = SI size`

The size of the file record does not match the search item length for an ODXTRANSFER.

ACTION: Create the file with a record size the same as the search item length.

`-802`	`Omnidex`	`File capacity is too small`

The capacity of the file for an ODXTRANSFER is too small for the number of search items to be transferred.

ACTION: Create the file with a larger capacity or use a new file name to let ODXTRANSFER specify the capacity.

`-803`	`Omnidex`	`Not an ODXTRANSFER ID file`

The file created for an external file search does not satisfy the following criteria:

Code- 7604; Size- 2 words; Type- Fixed Binary; Blocking Factor- 64

ACTION: Check the file in question with FUTIL, or a similar file utility and change the file to meet the above criteria, or requalify the IDs and specify a new file name as the destination.

Create a new file to hold the IDs.

`-804`	`Omnidex`	`Insufficient stack space`

An attempt was made to write Omnidex SIs or IDs to a new file during ODXTRANSFER, but there is not enough stack space for the large number of SIs or IDs.

ACTION: Try breaking the ODXFIND that preceded the ODXTRANSFER into smaller components. This would mean writing the results of your qualification to several external files, but they can be combined using an OR (+) operation.

`-805`	`Omnidex`	`Can't transfer SIs to an ID file`

A file that was previously created during an ODXTRANSFER Mode 201 has been specified as the destination for ODXTRANSFER Mode 1 or 2.

ACTION: Specify a new file name as the destination.

```
-806            Omnidex          Bad data item reference
```

The field passed through the last 16 bytes of the *Options* parameter during a Mode Option 100 ODXTRANSFER operation cannot be found.

ACTION: Check to see that the field exists in the data set referenced in the first 16 bytes of the *Options* parameter, and that the field name is spelled correctly.

```
-807            Omnidex          Bad data set reference.
```

The data set passed through the first 16 bytes of the *Options* parameter during a Mode Option 100 ODXTRANSFER operation cannot be found.

ACTION: Check to see that the data set exists in the data base referenced in the *Base* parameter, and that the set name is spelled correctly.

```
-808            Omnidex          Must be manual master or
                                 detail
```

A a Mode Option 100 ODXTRANSFER operation was attempted on an automatic master set.

ACTION: Automatic Masters are not supported by ODXTRANSFER Mode Option 100 operations. If you require the transfer of a key field to a file, perform an ODXFIND on the related detail set. Then, use ODXTRANSFER Mode Option 100 (if the key field is not the detail set's Omnidex SI) to transfer the key field.

```
-809            Omnidex          Detail chain processing not
                                 valid.
```

An attempt was made to transfer an SI field from a detail set in an SI domain via ODXTRANSFER Mode Option 100.

ACTION: Because this would require a chained read, ODXTRANSFER does not support such an operation. If a Multifind operation between a master in one domain and a detail in another is required, the first ODXFIND should be performed on the master set. The SIs can be transferred via an ODXFIND Mode 1 for a subsequent ODXFIND Multifind search on the detail in question. See the **Multifind** section in the **Data Base Design** chapter of the **OMNIDEX Administrator's Guide**, or the ODXFIND section of this chapter for more information about Multifind operations.

`-810`	`Omnidex`	`Last ODXFIND done on a DR domain`

An attempt was made to transfer SIs via ODXTRANSFER Mode 1 after an ODXFIND on a DR detail set.

ACTION: Because DR detail sets are indexed by IMAGE record number, ODXTRANSFER Mode 1 does not recognize their SIs. Use ODXTRANSFER Mode 101 to transfer SI fields from any DR details.

`-811`	`Omnidex`	`Unable to transfer field specified`

ODXTRANSFER was unable to retrieve, and therefore transfer, the records qualified by ODXFIND.

ACTION: Verify that the data base, data set and field are valid and that the data base exists and has been opened successfully.

If the above conditions are met, the problem may be in your index sets. Reindex the domain using ODXUTIL.

900s - DBIOPEN

`900`	`IMSAM`	`No IMSAM rootfile (IMSAM-ROOTFILE set is empty)`

The IMSAM rootfile is empty, therefore IMSAM processing cannot occur.

ACTION: Reinstall IMSAM on your data base using DBINSTAL.

See the **Installation** chapter for more information.

`901`	`IMSAM`	`Rootfile corrupt`

The IMSAM rootfile is corrupted.

ACTION: Call the DISC response center.

`903 (nc)`	`IMSAM`	`Deferred update must be completed using ODXUTIL`

An attempt was made to access a data base in write mode that was updated in deferred mode. An ODXUTIL UPDATE must be performed before the data base can be accessed again in write mode.

ACTION: Run ODXUTIL and perform an UPDATE on the data base or open the data base in Mode 103 or 104 to update the domain that was pending a deferred update.

`905`	`IMSAM`	`IMSAM has expired`

ACTION: If you have a trial version of IMSAM, you can order a new trial or a production version.

If you have a production version of IMSAM, install your updated IMSAM from the update tape you receive from DISC. Make sure that copies of the XL or SL are moved to every account that uses IMSAM.

`905`	`Omnidex`	`Omnidex has expired`

ACTION: If you have a trial version of Omnidex, you can order a production version. If this is a production version of Omnidex, you need to install the updated Omnidex tape you received from DISC.

`906`	`IMSAM`	`Insufficient stack space for the ILCB`

There is insufficient space in the stack for IMSAM to create the IMSAM local control block (ILCB).

ACTION: Rerun the program with a larger MAXDATA parameter, like 31,232, or specify Control Option 800 in the call to DBIOPEN. Control Option 800 uses an extra data segment for the IMSAM and Omnidex local control blocks.

`906`	`Omnidex`	`Insufficient stack space for the OLCB`

There is insufficient space in the stack for Omnidex to create the Omnidex local control block (OLCB).

ACTION: Rerun the program with a larger MAXDATA parameter, like 31,232, or specify Control Option 800 in the call to DBIOPEN. Control Option 800 uses an extra data segment for the IMSAM and Omnidex local control blocks.

907	IMSAM	IMSAM-ROOTFILE not accessible (bad password)

The IMSAM rootfile is not accessible because a bad password was used to open the data base. This can occur if some data sets in the data base have access class 0 (zero), but there is no password for that class.

ACTION: Use a valid password to access the data base.

908	IMSAM	IMSAM rootfile partially erased

The IMSAM rootfile has been partially erased, therefore IMSAM processing cannot occur.

ACTION: Reinstall IMSAM on your data base using DBINSTAL.

See the **Installation** chapter for more information.

909	IMSAM	LOADPROC failure

An attempt to perform a LOADPROC failed.

ACTION: Check the code segment table (CST), extended code segment table (XCST) and code segment block table (CSTBK) with Tuner or Opt. Increase the sizes if they are full. Also increase the size of the loader segment table with SYSDUMP if necessary.

909	Omnidex	MAXDATA too small

There is insufficient stack space available for the Omnidex intrinsics to function.

ACTION: Rerun the program with a larger MAXDATA parameter, like 31,000, or reduce the amount of stack space used by your program for forms, data declarations, etc.

913 (nc)	IMSAM	Index set is inaccessible

The index set is inaccessible, so the data base cannot be opened.

ACTION: Change the MPE security to enable access to the data base or release the data base using DBUTIL.

931	Omnidex	Cannot create file ODXUF0x

Omnidex is unable to create an ODXUF0 file for the deferred update. The file may already exist from another deferred update, or ODXUTIL INDEX, in the same group or from a previous deferred update, or INDEX, that aborted. Or, there may not be enough disk space.

ACTION: If an ODXUTIL INDEX or UPDATE is in process, wait until it completes before starting the next deferred update. If ODXUF0x files were left from an aborted process, purge them and then reindex the domain with an ODXUTIL INDEX. If you are short on disk space, store off some files or purge them.

932	Omnidex	Unload file BF/recsize is wrong

A file equation specified for an ODXUF0x file has the wrong blocking factor or record size.

ACTION: Specify only "DISC= *#recs*" parameter for an ODXUF0x file equation to let Omnidex create the file the correct blocking factor and record size.

946	IMSAM	Insufficient space in the extra data segment for the ILCB

Too many IMSAM and Omnidex data bases have been opened by the current process.

ACTION: Close some data bases or call the DISC response center.

949	Omnidex	Excluded words list is damaged

The excluded words list stored in the IMSAM rootfile and compared during an ODXFIND has been damaged.

ACTION: Run ODXUTIL and use the XCLUDE command to respecify the file containing the excluded words list.

951	IMSAM or Omnidex	Maximum number of extra data segments exceeded

The number of extra data segments requested by the program you are running exceeds the maximum per process allowed on your system.

ACTION: Reconfigure your system and increase the maximum number of data segments allowed per process. Omnidex requires one extra data segment per data base, and an additional data segment per process if you use the 800 Control Option in the DBIOPEN statement. (See DBIOPEN in the **Intrinsics** chapter of the **OMNIDEX Programmer's Guide** for information about this option.)

Increase the allowable number of data segments by the appropriate amount.

952 (nc)	IMSAM or Omnidex	Not enough virtual memory for Omnidex extra data segment

The amount of virtual memory required for Omnidex's extra data segment is not available to Omnidex.

ACTION: For a long-term solution to this problem, you need to do a system configuration and allocate 10 - 20% more space for virtual memory. A temporary solution is to lower the number of sessions running concurrently.

977	IMSAM	Not a Version 2.03 rootfile. Reinstall IMSAM

There were several changes in the internal layout of the IMSAM rootfile in Version 2.03.

ACTION: You must reinstall IMSAM on the data base using DBINSTAL.

991 (nc)	IMSAM	DL area contention with another subsystem

DL area is a special area of stack reserved for use by system software like screen processors, data base software, languages, etc. Occasionally, there is contention between Omnidex and other software that requires this space. When such contention occurs, this message is displayed.

ACTION: Omnidex uses the DL area for the Omnidex and IMSAM local control blocks by default. You can solve most DL contention problems by specifying Control Option 800 in the DBIOPEN call. This puts the OLCB and ILCB in an extra data segment. Still, words 11 and 12 of the DL-DB are used by OMNIDEX. See **DBIOPEN** for more information about Control Option 800. See the **Configuring OMNIDEX for Your Site** section of the **Appendix**.

```
-911          IMSAM          Option 100 requires exclusive
                             update access (Mode 3 or 4)
```

An attempt was made to open a data base with a base mode other than 3 or 4, but with Mode Option 100 added to it. Mode Option 100 can be used with Mode 3 or 4 only.

ACTION: Change the mode used to open the data base.

```
-912          IMSAM          Control Option 800 must be
                             used in the first DBIOPEN
```

Control Option 800 (specified in DBIOPEN by adding 800 to the IMAGE mode used) indicates that an extra data segment should be used for the IMSAM and Omnidex local control blocks. If any data bases are opened with this Control Option, all must be.

ACTION: Change all DBIOPENs to either specify Control Option 800,or not specify Control Option 800. (Call conversion uses Control Option 800 automatically.)

```
-914          IMSAM          call conversion library is
                             not a current version
```

The call conversion Library that you are using is not valid with the version of IMSAM that you are using.

ACTION: Reinstall the call conversion library from the production tape that contains your current version of IMSAM.

Chapter Overview

This chapter describes how to call the Omnidex and IMSAM intrinsics from within application programs.

The first section, **General Considerations**, discusses preparing programs, placing the OMNIDEX intrinsic library, and using the call conversion library to automatically convert IMAGE intrinsics to IMSAM and Omnidex intrinsics. Because the function and placement of XLs versus SLs differs between Version 2.06 and Version 2.05, we've provided a separate section for each version.

The second section, **Writing Programs**, discusses some general aspects of programming, including opening the data base, locking considerations, updating records and retrieving data by calling the Omnidex and IMSAM intrinsics. It also provides examples of 3GL applications through COBOL, and FORTRAN.

Other publications are available from DISC, which discuss in detail OMNIDEX applications written in a variety of programming languages. These can be obtained for free by contacting your DISC sales representative or DISC technical services.

Before writing programs to access or update OMNIDEX-enhanced data bases, programmers should be familiar with Omnidex's access capabilities. The **Intrinsics** chapter describes the syntax of the Omnidex and IMSAM intrinsics, as called through your application programs.

General Considerations

Introduction

There are some general considerations when developing applications that use Omnidex and IMSAM intrinsics, or for adapting any program to access a data base enhanced by OMNIDEX. They are:

- ❑ How to prepare the program
- ❑ How to minimize the stack size in an MPE V environment
- ❑ How to use the OMNIDEX intrinsics library
- ❑ How to use the call conversion library
- ❑ How to use switch stubs, if necessary

Preparing Programs

Stack Size

Version 2.05

When preparing programs that access the OMNIDEX Version 2.05 intrinsics, programmers should consider the stack size and program capabilities that are required.

☞ Note: The DISC account is created with Privileged Mode (PM) for OMNIDEX installation. Once OMNIDEX has been installed and the indexes have been populated, you can remove PM capability from the DISC account.

OMNIDEX uses the program's stack for the Omnidex Local Control Block (OLCB), the IMSAM Local Control Block (ILCB) and for Omnidex ID values during a call to the ODXFIND intrinsic. Therefore, it is necessary to prepare (prep) a program with a larger stack size than might normally be used.

It is difficult to predict how large of a stack size is required. Therefore, it is recommended that you prepare programs with a MAXDATA of 31232.

Account Capabilities

OMNIDEX uses extra data segments to store Omnidex IDs during calls to ODXFIND. The ILCB and OLCB can also reside in an extra data segment when a Control Option 800 DBIOPEN is issued. Control Option 800 can be issued explicitly or by running a program through call conversion.

Therefore, the account and group where the program(s) reside must have DS capability and you must prep any program that accesses an OMNIDEX-enhanced data base with DS capability.

For example:

```
:COBOL sourcefilename
:PREP $oldpass,programname;CAP=IA,BA,DS;MAXDATA=31232
```

With fourth generation languages, like QUICK and TRANSACT/3000, and also with Query, the program file is already prepared. You can use the FUTIL program to set the MAXDATA of the program and to add DS capability.

FUTIL is in the UTIL group of the DISC account. To run it, enter:

```
:RUN FUTIL.UTIL.DISC
```

The prompts and sample responses for MPE XL are:

```
Operation (Copy/Modify/Purge/Rename/Exit)  ? M

FILE NAME?  QUICK
CAPABILITIES [IA, BA ]  ?  DS, IA, BA

Operation (Copy/Modify/Purge/Rename/Exit)  ? E
```

Note that the current/default values are enclosed in brackets. The only difference is that with MPE XL you aren't prompted for MAXDATA.

The prompts and sample responses for MPE V are:

```
Operation (Copy/Modify/Purge/Rename/Exit)  ? M

FILE NAME?  Query
MAXDATA [15000]  ? 31232
CAPABILITIES [IA, BA ]  ?  DS, IA, BA

Operation (Copy/Modify/Purge/Rename/Exit)  ? E
```

Stack Size Considerations

Omnidex and IMSAM use a program's stack for the following purposes:

1. The Omnidex and IMSAM local control blocks reside in stack unless the data base is opened with Control Option 800.
2. Even if the data base is opened with Control Option 800, the OLCB and ILCB are moved into stack each time an Omnidex or IMSAM intrinsic requires access to the local control blocks.
3. The ODXFIND intrinsic uses stack to store Omnidex IDs during keyword qualification.
4. Each time an Omnidex or IMSAM intrinsic is called, the code segment for that intrinsic is read into stack for execution.

There are several ways to minimize the amount of stack space required by Omnidex and IMSAM. These are described next.

ODXFIND Stack Usage

ODXFIND places qualified IDs in the stack, or in an extra data segment if there is not enough room in stack. Therefore, programs should be prepared with a MAXDATA of 31232 and with DS capability, as described earlier.

DBIOPEN Control Option 800

Calling DBIOPEN with Control Option 800 stores the local control blocks (OLCB and ILCB) in an extra data segment until they are needed by a call to an Omnidex or IMSAM intrinsic. The area occupied by the local control blocks is released when control is returned to the calling program. The stack space conserved by this method is significant if you are opening more than one data base enhanced by Omnidex or IMSAM in one program. Each data base requires an OLCB and/or an ILCB.

Note: If one Omnidex or IMSAM data base is opened with Control Option 800, all the Omnidex and IMSAM data bases accessed within the program must be opened with Control Option 800.

PASCAL requires the use of Mode Option 800 if any HELP commands are to be used. COBOL 85 and FORTRAN 77 may require the use of Control Option 800 depending on the size of the program.

Using V/3000

Omnidex and IMSAM are entirely compatible with V/3000. V/3000, however, uses the same area of stack that Omnidex and IMSAM use for storage of the OLCB and ILCB. You can optimize use of the DL-DB area of stack by issuing a call to VOPENFORMF before issuing a call to DBIOPEN.

When an Omnidex or IMSAM data base is closed, the area of stack that the local control block(s) occupied is released. If the data base is opened before the formfile, the released area is not available to V/3000 because it resides above the area that V/3000 is using in stack.

Conversely, if the call to VOPENFORMF is issued before the DBIOPEN call, the area of stack occupied by the control block(s) is below the area occupied by V/3000. When a DBICLOSE releases the stack space used by the control blocks, V/3000 can use the newly released area.

This technique is especially useful in programs that sequentially open and close multiple Omnidex or IMSAM enhanced data bases.

Using SORT/3000

It is recommended that input and output files be used, whenever possible, within programs that call Omnidex and IMSAM intrinsics. This reduces the amount of stack used by SORT operations, and allows Omnidex and IMSAM to work freely in a limited stack environment.

Configuring OMNIDEX for Your Site

There are two means to configuring the OMNIDEX intrinsics to your systems resources:

- ❑ The DBINSTAL CONFIG command.
- ❑ DBINSTAL entry points .

Both means enable you to accommodate your system's stack and disk space when running OMNIDEX. They enable you to configure:

- ❑ The location in the DL-DB area of stack where the local control blocks (OLCB and ILCB) reside.
- ❑ The maximum size of the extra data segment created for a DBIOPEN Mode 800 open.
- ❑ The size of the session-temporary ODXFIND ID files that are used to hold Omnidex IDs during an Omnidex search.
- ❑ The amount of headroom allocated for other intrinsics calls within ODXFIND.

The DBINSTAL CONFIG command and the DBINSTAL entry points are discussed in the **Installation** chapter of the **OMNIDEX Administrator's Guide**.

OMNIDEX Version 2.06 XL and SL Placement

The placement of XLs and SLs for OMNIDEX Version 2.06 is discussed in the following pages. Whenever several alternative methods of copying XLs and SLs are listed, the recommended method has a check mark next to it (✓).

The illustration on the following page is a matrix that shows the flow of calls when OMNIDEX Version 2.06 is run:

- In Native Mode
- In Compatibility Mode
- With call conversion
- Without call conversion

The OMNIDEX Intrinsic Library

Applications

The OMNIDEX intrinsic library resides in the PUB group of the DISC account. The library, for Version 2.06, is an executable library (XL.PUB.DISC) .

The OMNIDEX intrinsic library contains the following procedures for all the intrinsics:

DBIOPEN	DBICLOSE	DBIFIND
DBIGET	DBIPUT	DBIDELETE
DBIUPDATE	DBILOCK	DBIUNLOCK
DBIERROR	DBIEXPLAIN	DBIINFO
ODXFIND	ODXGET	ODXGETWORD
ODXINFO	ODXPRINT	ODXVIEW
	ODXTRANSFER	

In addition, it contains internal routines used by the intrinsics.

Note: DISC SLs, USLs, and XLs are Version 2.06 in these illustrations.

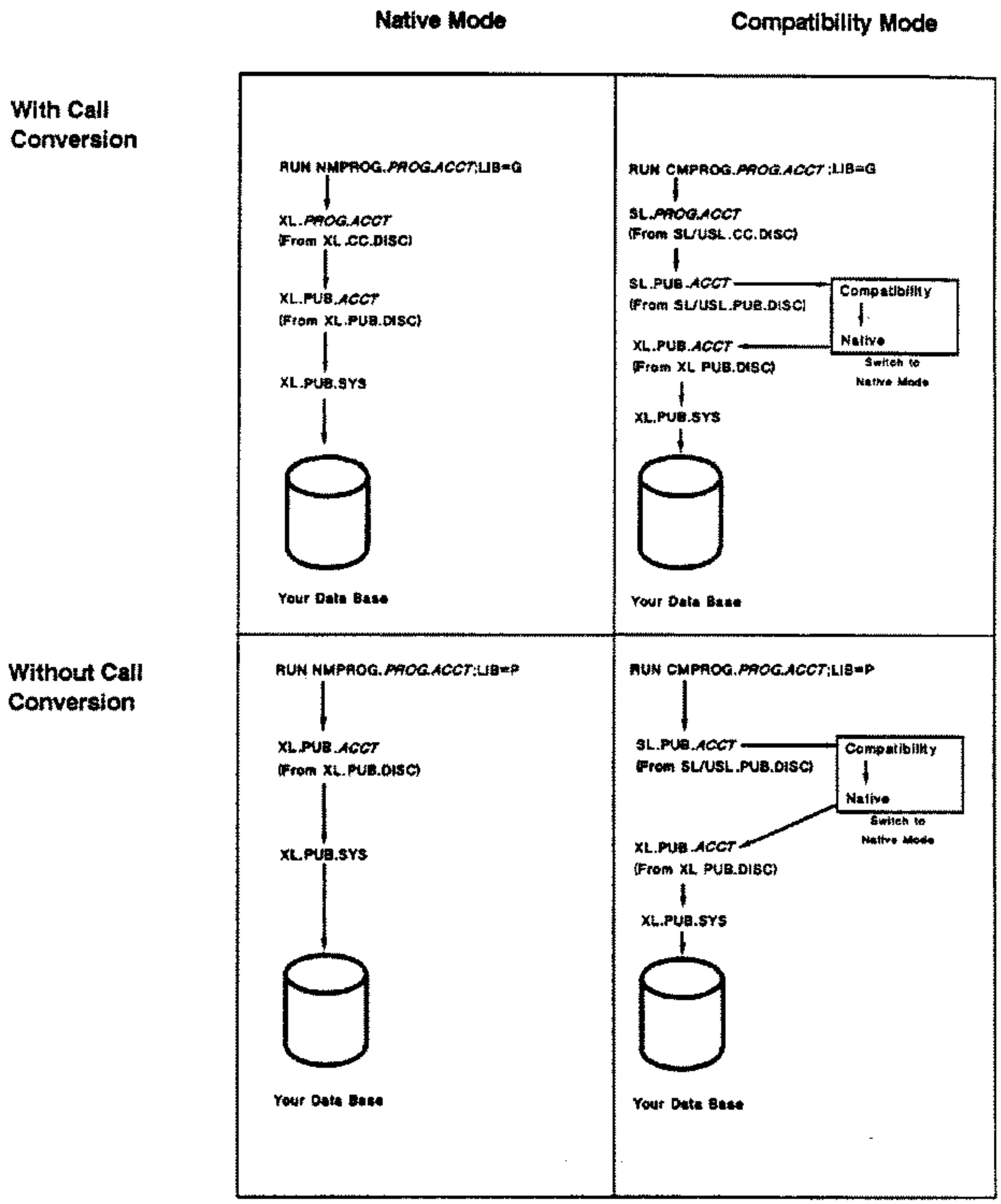


Figure 3 - 1: The OMNIDEX 2.06 flow of calls

The intrinsic library supports Omnidex and IMSAM retrieval on a data base enhanced by OMNIDEX. Whenever an Omnidex or IMSAM intrinsic is called by an application program run with the appropriate **LIB=** parameter, the appropriate intrinsic in the intrinsic library is accessed.

These intrinsics, in turn, call the IMAGE intrinsics in the system level intrinsic library to perform IMAGE retrieval and update functions.

Calling the Executable Intrinsic Libraries

For a program to access the Omnidex and IMSAM intrinsics, the intrinsic library must reside in the same group as the program, or in the PUB group of the account where the program resides.

The flow of control is the same whether the intrinsic library is in the same group as the application programs or in the PUB group of the account that contains the programs. The program calls the intrinsics in the XL and they, in turn, call IMAGE intrinsics in the System intrinsic library (XL.PUB.SYS).

For example, if the intrinsic library is in the program account, the flow of control for a call to DBIOPEN is:

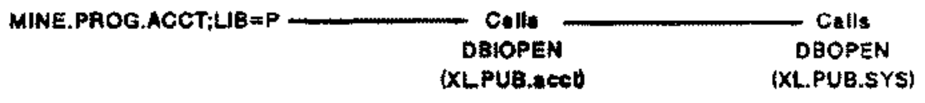


Figure 3 - 2: The flow of calls when the intrinsic library resides in the program account

If the programs are moved to the DISC account, the flow of control is:

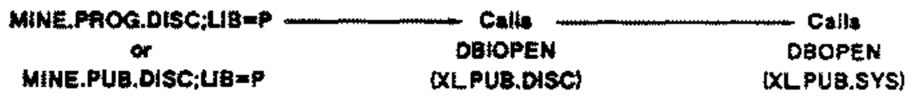


Figure 3 - 3: The flow of calls when the intrinsic library resides in PUB.DISC

You can copy XL.PUB.DISC to the account where your programs reside, then run the programs as follows:

❑ If the XL is copied to the same group as the programs, the programs must be run with LIB=G.

or

☑ If the XL is copied to the PUB group of the programs' account, the programs are run with LIB=P. (Notice that this is the PUB group of the account that contains the programs, not the user's log-on account.)

If call conversion is used, the call conversion XL must be placed in the programs' group and the intrinsic XL must be placed in the PUB group. See the section on the **Conversion of IMAGE Calls**, which follows, for more information.

If call conversion is not used, you can copy XL.PUB.DISC to either the group where your application programs reside or to the PUB group of the account where they reside.

For example, if you have a program called MINE that resides in the PROG group of a production account called PRODUCT, you could do either of the following:

❑ Copy XL.PUB.DISC to XL.PROG.PRODUCT
Run the program with LIB=G

```
(:RUN MINE.PROG.PRODUCT;LIB=G)
```

or

☑ Copy XL.PUB.DISC to XL.PUB.PRODUCT
Run the program with LIB=P

```
(:RUN MINE.PROG.PRODUCT;LIB=P)
```

Then, when an Omnidex or IMSAM intrinsic is called from a program, it accesses the appropriate procedure in the intrinsic library (either XL.*group* or XL.PUB).

If you prefer, you can move your application programs to the DISC account instead of copying the intrinsic library to your application programs' account. You could create a new group for the programs or copy them to the PUB group.

❑ If you copy the programs to a separate group in the DISC account, run them with LIB=P.

or

❑ If you copy them to the PUB group of the DISC account, run them with LIB=P or LIB=G.

Moving or Copying Executable Libraries

If you already have an XL in the same group as your application program and in the PUB group of the program account, you can either combine the OMNIDEX intrinsic library with yours, or use the **XL=** parameter in the RUN statement.

To copy the OMNIDEX intrinsic library XL into your XL, perform the following steps:

1. Run LINKEDIT by typing the **LINKEDIT** command:

   ```
   :LINKEDIT
   ```

2. Copy the OMNIDEX intrinsic library XL into your XL using the COPYXL command:

   ```
   LinkEd> COPYXL FROM=XL.PUB.DISC;TO=XL.PUB.acct
   ```

3. Exit LINKEDIT

   ```
   LinkEd>  EXIT
   ```

To specify the XL parameter in a RUN command, perform the following steps:

1. perform the following COPY operations at an MPE (:) prompt:

   ```
   :COPY XL.PUB.DISC,XLODX.PUB.PRODUCT
   ```

2. When you run your application programs be sure to specify the following XL= parameter:

   ```
   RUN
   myprog.PROG.PRODUCT;XL="XLODX.PUB.PRODUCT,XL.PUB.PRODUCT"
   ```

Note that the XLs specified in the **XL=** parameter is searched in the same order as they appear.

The Conversion of IMAGE Calls

The call conversion library resides in the CC group of the DISC account. This file, XL.CC.DISC, has two purposes:

1. **It facilitates the interface between OMNIDEX and fourth generation languages (4GLs).**
2. **It also intercepts IMAGE calls and redirects them to the appropriate IMSAM intrinsics without any program changes.** This means the Omnidex and IMSAM index sets are automatically updated by calls to DBPUT, DBDELETE and DBUPDATE.

Call conversion should be used whenever you add Omnidex or IMSAM retrieval capabilities to a data base and want to run your existing programs just as they are.

If the data base structure was not altered before installing OMNIDEX, the only change you may need to make to your programs is to use an @ item list, or the equivalent, for DBPUTs and for DBUPDATEs if the data base is opened in Mode 3. This is necessary for most sets enhanced with Omnidex or IMSAM. Master sets that use a transparent ID and DR detail sets are the exceptions.

If there were changes in the data set structure, like the addition of a J2 field to a master, you must modify the record definition in your 3GL program or 4GL data dictionary to reflect that change.

To use the advanced retrieval capabilities of Omnidex and IMSAM, you must add or change programs to call the Omnidex and IMSAM retrieval intrinsics, as well as DBIERROR and DBIEXPLAIN, and change the *Status* array to 21 16-bit-words. Note that you can still run the program under call conversion to intercept IMAGE calls and convert them to their corresponding IMSAM intrinsic calls.

Applications

The call conversion library intercepts calls to:

DBOPEN	DBPUT
DBCLOSE	DBDELETE
DBLOCK	DBUPDATE
DBUNLOCK	DBFIND
DBEXPLAIN	DBGET
DBERROR	DBINFO

It substitutes calls to the corresponding IMSAM intrinsics as necessary.

By accessing the call conversion library (using the **LIB=** parameter in the program RUN statement), an existing IMAGE application program may access any combination of IMAGE, Omnidex and IMSAM data bases.

When a program uses call conversion, calls to IMAGE intrinsics are trapped and converted to equivalent IMSAM intrinsic calls. For example, a Mode 1 call to DBPUT would be intercepted and converted to a Mode 1 call to DBIPUT. This enables the Omnidex and IMSAM indexes to be maintained automatically.

Installation and Function

For a program to make use of the call conversion library, the library (XL.CC.DISC) must reside in the same group as the program and the intrinsic library (XL.PUB.DISC) must reside in the PUB group, as follows:

- XL.CC is in the same group as the program.
- XL.PUB is in the PUB group of the programs account.
- The program is run with LIB=G.

These libraries must reside in the account that contains the application programs, not the user's log-on account.

Using LIB=G means the program first searches the group library when a call to an external intrinsic is issued. Then, the program searches the library in the PUB group, and finally the system library, until the call is satisfied.

This hierarchical search to satisfy calls to external procedures determines the placement of the libraries. The call conversion library must intercept calls to the IMAGE intrinsics, then, call the IMSAM intrinsics in the PUB library and still call the IMAGE intrinsics at the system level if required.

For example, if the program MINE resides in the PROG group of the PRODUCT account, you would copy the libraries as follows:

1. Copy the call conversion library to the same group as the program:

   ```
   XL.CC.DISC  ———>  XL.PROG.PRODUCT
   ```

2. Copy the intrinsic library to the PUB group of the account that contains the program:

   ```
   XL.PUB.DISC ———>  XL.PUB.PRODUCT
   ```

3. Run the program with LIB=G so that the IMAGE intrinsics is converted to IMSAM intrinsics:

   ```
   :RUN MINE.PROG.PRODUCT;LIB=G
   ```

Now, when a call to an IMAGE intrinsic, like DBOPEN, is issued:

1. The program accesses the intrinsic (DBOPEN) in XL.PROG.PRODUCT. This DBOPEN is merely an entry point that calls DBIOPEN.
2. The DBOPEN intrinsic in XL.PROG.PRODUCT calls the DBIOPEN intrinsic in XL.PUB.PRODUCT.
3. The DBIOPEN intrinsic issues a call to the IMAGE intrinsic DBOPEN, which resides in XL.PUB.SYS, then performs Omnidex and IMSAM housekeeping functions.

The flow of control works like this:

$$\begin{tikzcd}
\text{MINE.PROG.PRODUCT;LIB=G} \arrow[r] & \begin{array}{c}\text{Calls}\\ \text{DBOPEN}\\ \text{(XL.PROG.PRODUCT)}\end{array} \arrow[r] & \begin{array}{c}\text{Calls}\\ \text{DBIOPEN}\\ \text{(XL.PUB.PRODUCT)}\end{array} \arrow[r] & \begin{array}{c}\text{Calls}\\ \text{DBOPEN}\\ \text{(XL.PUB.SYS)}\end{array}
\end{tikzcd}$$

Figure 3 - 4: The flow of control when libraries are copied to the program account

If you prefer, you can move your application programs to the CC group of the DISC account instead of moving the libraries. If you do, run the programs as follows:

```
:RUN MINE.CC.DISC;LIB=G
```

Because the libraries are placed correctly in the DISC account, calls to IMAGE intrinsics are intercepted by XL.CC.DISC, which call the corresponding IMSAM intrinsics in XL.PUB.DISC, which in turn call the IMAGE intrinsics in XL.PUB.SYS.

When you move the programs to the CC group of the DISC account instead, the flow of control is:

$$\begin{tikzcd}
\text{MINE.CC.DISC;LIB=G} \arrow[r] & \begin{array}{c}\text{Calls}\\ \text{DBOPEN}\\ \text{(XL.CC.DISC)}\end{array} \arrow[r] & \begin{array}{c}\text{Calls}\\ \text{DBIOPEN}\\ \text{(XL.PUB.DISC)}\end{array} \arrow[r] & \begin{array}{c}\text{Calls}\\ \text{DBOPEN}\\ \text{(XL.PUB.SYS)}\end{array}
\end{tikzcd}$$

Figure 3 - 5: The flow of control when programs are copied to CC.DISC

You could copy the call conversion XL modules (XL.CC.DISC) to XL.PUB of the program account and add the intrinsic XL modules (XL.PUB.DISC) to XL.PUB.SYS using LINKEDIT.

Note: Because XL.PUB.SYS is usually being accessed, you might copy it to a different name (like, XLC.PUB.SYS). Add the intrinsic XL modules (XL.PUB.DISC) to this copy. You must then create a configuration (COLDLOAD) tape and use it to update the operating system.

Using Call Conversion with Existing XLs

If you already have an XL in the same group as your application program, and in the PUB group of the account that contains the programs, you can either combine the call conversion library with yours or use the **XL=** parameter in the RUN statement.

To copy the OMNIDEX intrinsic library XL into your XL, perform the following steps:

1. Run LINKEDIT by typing the **LINKEDIT** command:

```
:LINKEDIT
```

2. Copy the call conversion library XL into your XL using the COPYXL command:

```
LinkEd> COPYXL FROM=XL.CC.DISC;TO=XL.prog.acct
```

3. Exit LINKEDIT

```
LinkEd> EXIT
```

To specify the XL parameter in a RUN command, perform the following steps:

1. Perform the following COPY operations at an MPE (:) prompt:

```
:COPY XL.CC.DISC,XLODX.PROG.PRODUCT
:COPY XL.PUB.DISC,XLODX.PUB.PRODUCT
```

2. When you run your application programs be sure to specify the following XL= parameter:

```
RUN myprog.PROG.PRODUCT;XL="XLODX.PROG.PRODUCT, &
XL.PROG.PRODUCT,XLODX.PUB.PRODUCT,XL.PUB.PRODUCT"
```

Note that the XLs specified in the **XL=** parameter are searched in the same order as they appear.

Additional Notes

Explicit calls to the IMSAM intrinsics may be intermixed with implicit calls to IMSAM intrinsics made by programs under call conversion.

For example, if one of your existing programs is being run under call conversion, you can add calls to the Omnidex and IMSAM intrinsics to use the new retrieval capabilities. You can still run the program under call conversion to handle any IMAGE calls.

Another example is that of a vendor-supplied program for which you can write procedures to be executed while the program is running. With this type of program, you can use call conversion and add procedures with calls to the Omnidex and IMSAM intrinsics.

The call conversion library generates IMSAM intrinsic calls in response to certain IMAGE calls by the program, while the user-written procedures call the IMSAM intrinsics directly, as needed.

When using call conversion, the IMSAM retrieval intrinsics DBIFIND and DBIGET also can be called indirectly. The call conversion library converts calls for DBFIND and DBGET to DBIFIND and DBIGET, passing all the parameters unchanged, except the *Status* array.

This means a Mode 7 DBGET is converted to a Mode 7 DBIGET, which is performed just like a DBGET Mode 7. This also means a Mode -104 DBGET (an illegal mode in IMAGE) is converted to a Mode -104 DBIGET, which enables you to do an IMSAM partial-key retrieval with a 4-byte key value.

In short, you can do IMSAM retrievals using IMAGE intrinsics by specifying IMSAM modes for DBGET and running the program under call conversion.

If calls to IMAGE intrinsics are mixed with calls to IMSAM intrinsics, then the program's error handling routines must be modified to call the appropriate intrinsics (DBERROR for IMAGE calls, DBIERROR for IMSAM calls). For example, if a program calls DBGET under call conversion, then DBERROR or DBEXPLAIN should be called if the call to DBGET fails. Conversely, if a program calls DBIGET, then DBIERROR or DBIEXPLAIN should be called if the call to DBIGET fails.

OMNIDEX Intrinsic Switch Stubs

Both versions of OMNIDEX can be called by either Native Mode or Compatibility Mode application programs. This is possible because of *switch stubs*, library routines that convert Compatibility Mode calls to Native Mode calls, and vice versa.

If you plan to call OMNIDEX Version 2.06 intrinsics from a Compatibility Mode program, OMNIDEX includes a switch stub Segmented Library file called SL.PUB.DISC. It is a segmented library that functions as an interface between your Compatibility Mode programs and the OMNIDEX Native Mode intrinsic library (XL.PUB.DISC).

Thus, if you are running any Compatibility Mode programs, you should copy the switch stub library (SL.PUB.DISC), and the OMNIDEX intrinsic XL (XL.PUB.DISC), to the PUB group of the account that contains the program.

Note: If you are upgrading from the Compatibility Mode version, Version 2.05, of OMNIDEX, you must purge the existing Version 2.05 intrinsic library segments from the program accounts' PUB group before adding the Version 2.06 switch stub segments. The Segmenter PURGE syntax is:

```
          PURGESL SEGMENT,SEGMENT'NAME
```

The segments you must purge are DISC'1, DISC'2, DISC'3, DISC'4, DISC'5, DISC'6, DISC'7, DISC_C, CCSCSEG0, CCSCSEG1, and CCSCSEG2.

The SL Segments can be copied using Hewlett-Packard's Segmenter utility as follows:

```
:SEGMENTER

-SL SL.PUB.PRODUCT
-USL SL.PUB.DISC
-ADDSL DISC'2
-ADDSL DISC'CM'SWITCH
-ADDSL DISC'5
-ADDSL DISC'6
-ADDSL DISC'7
-ADDSL DISC_C
-ADDSL CCSCSEG0
-ADDSL CCSCSEG1
-ADDSL CCSCSEG2
-EXIT
```

You must copy any software updates into the PUB group of the program account to use the switch stubs.

To use the OMNIDEX 2.06 switch stubs, run your Compatibility Mode programs with the **LIB=P** option. MPE detects that your program is running in Compatibility Mode and loads the SL switch stubs which, in turn, call the XL routines in the PUB group.

For example, you may have a Compatibility Mode program that directly calls Omnidex or IMSAM intrinsics on an OMNIDEX-enhanced data base. The flow of control when the program is run would be as follows:

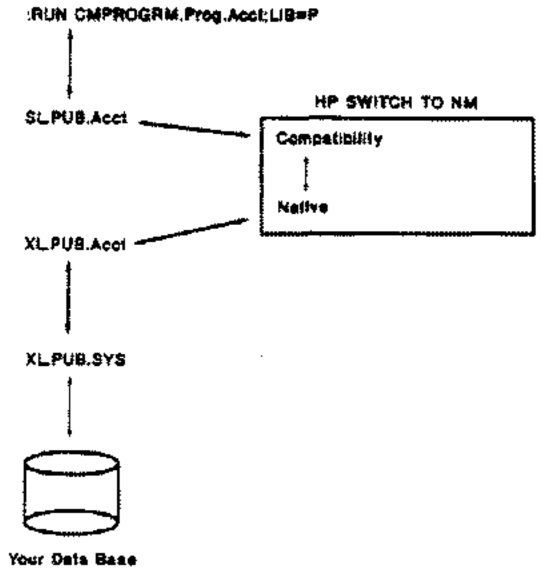


Figure 3 - 6: Calling OMNIDEX 2.06 from Compatibility Mode

Using Call Conversion with Intrinsic Switch Stubs

To run a Compatibility Mode program under call conversion in Version 2.06, you must use the call conversion routines in SL.CC.DISC and the intrinsic switch stubs in SL.PUB.DISC.

To begin using call conversion, copy SL.CC.DISC into the group and account where the program resides. If you already have a group SL for your application, you can copy the required segments from USL.CC.DISC using the following Segmenter commands:

```
:SEGMENTER

-SL SL.group.product
-USL USL.CC.DISC
-ADDSL IMAGE'CC
-ADDSL TIMAGE11  <for BASIC only>
-EXIT
```

Note: If you are upgrading from the Compatibility Mode version, Version 2.05, of OMNIDEX, you must purge the existing Version 2.05 call conversion library segments from the group and account where the program resides, before adding the Version 2.06 call conversion segments. The Segmenter PURGE syntax is:

```
PURGESL SEGMENT,SEGMENT'NAME
```

The segments you must purge are IMAGE'CC, FPROC'CC and TIMAGE11.

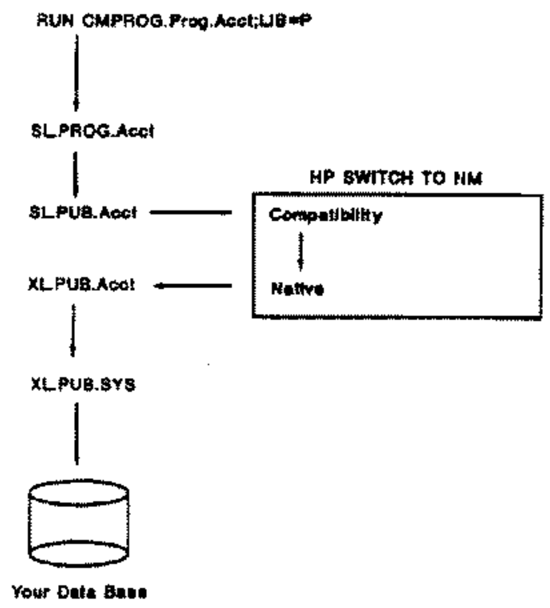


Figure 3 - 7: When call conversion switch stubs are copied to the program account

If you are running Native Mode programs from the same account as Compatibility Mode programs (shown in **Figure 3 - 1**) you must perform the following steps:

- ❑ Follow the instructions in this section to either copy the necessary call conversion and intrinsic switch stubs to your application group or copy the necessary procedure segments to existing SLs in your application group.
- ❑ Refer to **The Conversion of IMAGE Calls** section and follow the instructions for copying XL.CC.DISC to your application group and account.

Note: OMNIQUIZ Version 2.06 must be run with a Native Mode version of QUIZ on the 900 series machines.

OMNIDEX Version 2.05 SL and XL Placement

The placement of SLs and XLs for OMNIDEX Version 2.05 is discussed in the following pages. Whenever several alternative methods of copying XLs and SLs are listed, the recommended method is marked with a check (✓).

The OMNIDEX Version 2.05 Intrinsics Library

The OMNIDEX intrinsic library contains the following procedures for all the intrinsics:

DBIOPEN	DBICLOSE	DBILOCK
DBIUNLOCK	DBIFIND	DBIGET
DBIPUT	DBIDELETE	DBIUPDATE
DBIERROR	DBIEXPLAIN	DBIINFO
ODXFIND	ODXGET	ODXGETWORD
ODXINFO	ODXPRINT	ODXVIEW
	ODXTRANSFER	

In addition, it contains internal routines used by the intrinsics.

The intrinsic library supports Omnidex and IMSAM retrieval on a data base enhanced by OMNIDEX. Whenever an Omnidex or IMSAM intrinsic is called by an application program run with the appropriate **LIB=** parameter, the appropriate intrinsic in the intrinsic library is accessed.

These intrinsics, in turn, call the IMAGE intrinsics in the system level intrinsic library to perform IMAGE retrieval and update functions.

Note: DISC SLs, USLs, and XLs are Version 2.05 in these illustrations.

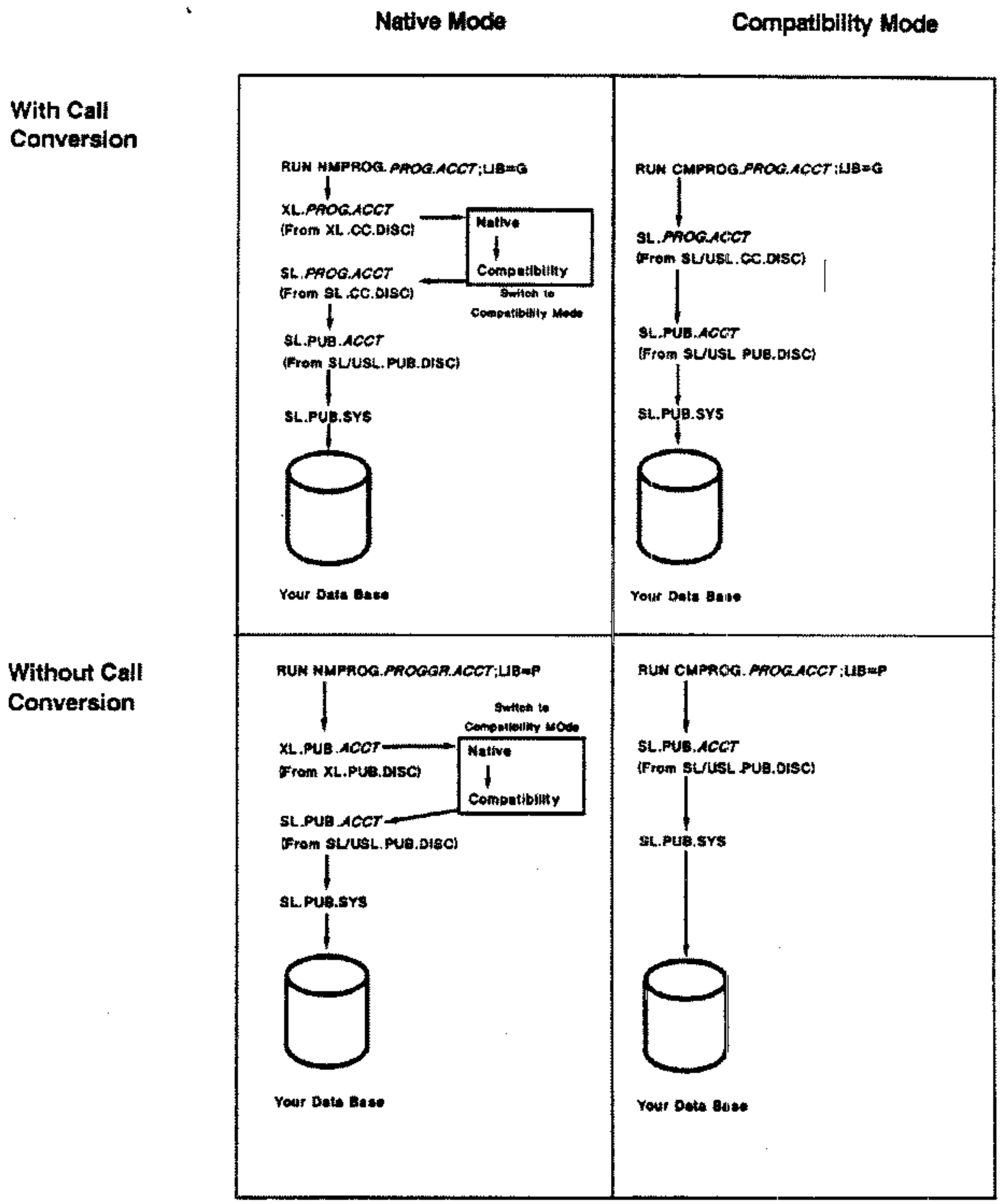


Figure 3 - 8: The OMNIDEX 2.05 flow of calls

The OMNIDEX intrinsic library for Version 2.05 is a segmented library (SL.PUB.DISC). A user subprogram library (USL) is also provided.

Calling from a Segmented Library

For a program to access the Omnidex and IMSAM intrinsics, the intrinsic library must reside in the same group as the program or in the PUB group of the account where the program resides.

The flow of control is the same whether the intrinsic library is in the same group as the application programs or in the PUB group of the account that contains the programs. The program calls the intrinsics in the SL and they, in turn, call IMAGE intrinsics residing in SL.PUB.SYS.

For example, if the intrinsic library is in the program account, the flow of control for a call to DBIOPEN is:

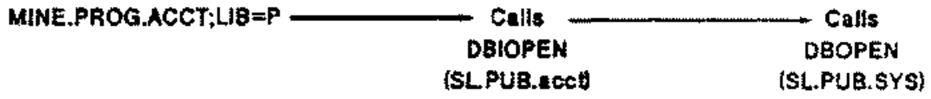


Figure 3 - 9: The flow of calls when the intrinsic library resides in the program account

If the programs are moved to the DISC account, the flow of control is:

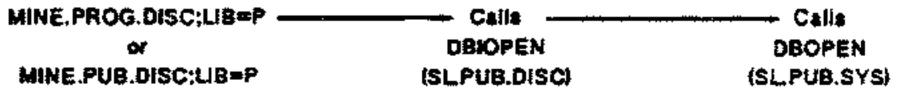


Figure 3 - 10: The flow of calls when the intrinsic library resides in PUB.DISC

You can copy the SL to the account where your programs reside, then run the programs as follows:

☐ If the SL is copied to the same group as the programs, the programs must be run with LIB=G.

or

☑ If the SL is copied to the PUB group of the programs' account, the programs should be run with LIB=P. (Notice that this is the PUB group of the account that contains the programs, not the user's log-on account.)

If you are using call conversion, the call conversion SL must be placed in the programs' group and the intrinsic library must be placed in the PUB group of the programs' account. See the section on **The Conversion of IMAGE Calls**, which follows, for more information.

If you are not using call conversion, you can copy SL.PUB.DISC to either the group where your application programs reside or to the PUB group of the account where they reside.

For example, if you have a program called MINE that resides in the PROG group of a production account called PRODUCT, you could do either of the following:

❑ Copy SL.PUB.DISC to SL.PROG.PRODUCT
Run the program with LIB=G

```
(:RUN MINE.PROG.PRODUCT;LIB=G)
```

or

☑ Copy SL.PUB.DISC to SL.PUB.PRODUCT
Run the program with LIB=P

```
(:RUN MINE.PROG.PRODUCT;LIB=P)
```

Then, when an Omnidex or IMSAM intrinsic is called from a program, it accesses the appropriate procedure in the intrinsic library (either SL.group or SL.PUB).

If you prefer, you can move your application programs to the DISC account instead of copying the intrinsic library to your application programs' account. You could create a new group for the programs or copy them to the PUB group.

❑ If you copy the programs to a separate group in the DISC account, run them with LIB=P.

or

❑ If you copy them to the PUB group of the DISC account, run them with LIB=P or LIB=G.

User Subprogram Library for Version 2.05

If you already have an SL in the same group as your application program and in the PUB group of the account that contains the programs, you must combine the OMNIDEX intrinsics that are contained in the User Subprogram Library (USL.PUB.DISC) into your existing SL.

Use Hewlett-Packard's Segmenter to add the OMNIDEX segments to the existing SL in the program group.

For example:

```
:SEGMENTER

-SL SL.PROG.PRODUCT
-USL USL.PUB.DISC
-ADDSL DISC'1
-ADDSL DISC'2
-ADDSL DISC'3
-ADDSL DISC'4
-ADDSL DISC'5
-ADDSL DISC'6
-ADDSL DISC'7
-ADDSL DISC_C
-ADDSL CCSCSEG0
-ADDSL CCSCSEG1
-ADDSL CCSCSEG2
-EXIT
```

Then, run your programs as follows:

```
:RUN MINE.PROG.PRODUCT;LIB=G
```

or

```
:RUN MINE.PROG.PRODUCT;LIB=P
```

The Conversion of IMAGE Calls

The call conversion library resides in the CC group of the DISC account. This file, SL.CC.DISC, has two purposes:

1. **It facilitates the interface between OMNIDEX and fourth generation languages (4GLs).**
2. **It also intercepts IMAGE calls and redirects them to the appropriate IMSAM intrinsics without any program changes.** This means the Omnidex and IMSAM index sets are automatically updated by calls to DBPUT, DBDELETE and DBUPDATE.

Call conversion should be used whenever you add Omnidex or IMSAM retrieval capabilities to a data base and want to run your existing programs just as they are.

As long as no changes were made to the data base structure, the only changes that may be required in your programs are to use an @ item list, or the equivalent for DBPUTs and for DBUPDATEs, if the data base is opened in Mode 3. This is necessary for most sets enhanced with Omnidex or IMSAM. Master sets that use transparent ID and DR detail sets are exempt from this.

If there were changes in the data set structure, like the addition of a J2 field to a master, you must modify the record definition in your 3GL program or 4GL data dictionary to reflect that change.

To use the advanced retrieval capabilities of Omnidex and IMSAM, you must add or change programs to call the Omnidex and IMSAM retrieval intrinsics, as well as DBIERROR and DBIEXPLAIN, and change the *Status* array to 21 16-bit-words. Note that you can still run the program under call conversion to intercept IMAGE calls and convert them to their corresponding IMSAM intrinsic calls.

Applications

The call conversion library intercepts calls to:

DBOPEN	DBPUT
DBCLOSE	DBDELETE
DBLOCK	DBUPDATE
DBUNLOCK	DBFIND
DBEXPLAIN	DBGET
DBERROR	DBINFO

It substitutes calls to the corresponding IMSAM intrinsics as necessary.

By accessing the call conversion library (using the **LIB=** parameter in the program RUN statement), an existing IMAGE application program may access any combination of IMAGE, Omnidex and IMSAM data bases.

When a program uses call conversion, calls to IMAGE intrinsics are trapped and converted to equivalent IMSAM intrinsic calls. For example, a Mode 1 call to DBPUT would be intercepted and converted to a Mode 1 call to DBIPUT. This enables the Omnidex and IMSAM indexes to be maintained automatically.

For Version 2.05, the call conversion library opens a data base with DBIOPEN Control Option 800 and maintains the Omnidex and IMSAM local control blocks (OLCBs and ILCBs) in an extra data segment. This reduces contention for stack space between the local control blocks and your application programs.

Installation and Function

For a program to make use of the call conversion library, the library (SL.CC.DISC) must reside in the same group as the program and the intrinsic library (SL.PUB.DISC) must reside in the PUB group, as follows:

- ❑ SL.CC is in the same group as the program.
- ❑ SL.PUB is in the PUB group of the program's account.
- ❑ The program is run with LIB=G.

These libraries must reside in the account that contains the application programs, not the user's log-on account.

Using LIB=G means the program first searches the group library when a call to an external intrinsic is issued. Then, the program searches the library in the PUB group, and finally the system library, until the call is satisfied.

This hierarchical search to satisfy calls to external procedures determines the placement of the libraries. The call conversion library must intercept calls to the IMAGE intrinsics, then call the IMSAM intrinsics in the PUB library, and still be able to call the IMAGE intrinsics at the system level.

For example, if the program MINE resides in the PROG group of the PRODUCT account, you would copy the libraries as follows:

1. Copy the call conversion library to the same group as the program:

   ```
   SL.CC.DISC  ———>  SL.PROG.PRODUCT
   ```

2. Copy the intrinsic library to the PUB group of the account that contains the program:

   ```
   SL.PUB.DISC ———>  SL.PUB.PRODUCT
   ```

3. Run the program with LIB=G so that the IMAGE intrinsics are converted to IMSAM intrinsics:

   ```
   :RUN MINE.PROG.PRODUCT;LIB=G
   ```

Now, when a call to an IMAGE intrinsic, like DBOPEN, is issued:

1. The program accesses the intrinsic (DBOPEN) in SL.PROG.PRODUCT. This DBOPEN is merely an entry point that calls DBIOPEN.
2. The DBOPEN intrinsic in SL.PROG.PRODUCT calls the DBIOPEN intrinsic in SL.PUB.PRODUCT.
3. The DBIOPEN intrinsic issues a call to the IMAGE intrinsic DBOPEN, which resides in SL.PUB.SYS, then performs Omnidex and IMSAM housekeeping functions.

The flow of control works like this:

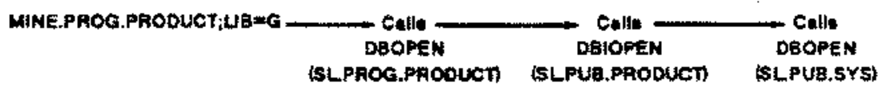

MINE.PROG.PRODUCT;LIB=G
Calls DBOPEN (SL.PROG.PRODUCT)
Calls DBIOPEN (SL.PUB.PRODUCT)
Calls DBOPEN (SL.PUB.SYS)

Figure 3 - 11: The flow of control when libraries are copied to the program account

If you prefer, you can move your application programs to the CC group of the DISC account instead of moving the libraries. If you do, run the programs as follows:

```
:RUN MINE.CC.DISC;LIB=G
```

Because the libraries are placed correctly in the DISC account, calls to IMAGE intrinsics are intercepted by SL.CC.DISC, which call the corresponding IMSAM intrinsics in SL.PUB.DISC, which in turn call the IMAGE intrinsics in SL.PUB.SYS.

If you choose to move the programs to the CC group of the DISC account instead, the flow of control is:

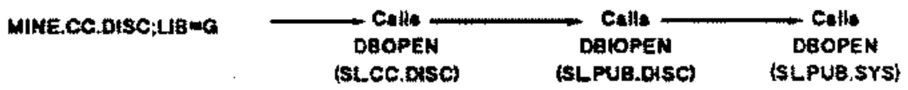

MINE.CC.DISC;LIB=G
Calls DBOPEN (SL.CC.DISC)
Calls DBIOPEN (SL.PUB.DISC)
Calls DBOPEN (SL.PUB.SYS)

Figure 3 - 12: The flow of control when programs are copied to CC.DISC

Installing the OMNIDEX Intrinsics in Your System SL

Although it is not recommended, you could add the call conversion USL segments to SL.PUB of the program account and add the intrinsics USL segments to SL.PUB.SYS. This might be necessary if your application programs must be run from the PUB group.

Although it is not recommended, you can add the User Subprogram Library (USL) to the system SL if it is necessary for your system. If you do this, you should create a new configuration (COLDLOAD) tape.

Here, the flow of control is:

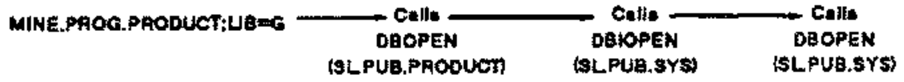


Figure 3 - 13: The flow of control when libraries are copied to PUB.SYS

WARNING! Do not add the segments from USL.CC.DISC and USL.PUB.DISC to the same SL. This causes an infinite loop. Calls to DBOPEN call DBIOPEN, which calls DBOPEN in the same SL. This would again call DBIOPEN, etc.

The program first accesses the SL in PUB.PRODUCT, where the call conversion intrinsics reside. Then, the IMSAM, and IMAGE, intrinsics are called from the SL.PUB.SYS.

Note: The following procedure will lock the system SL for the time it takes to add the new segment, and should be performed during a period of low system activity.

Use Hewlett-Packard's Segmenter to add the procedure segments within these USLs to the SLs in the PUB group of the program account and to PUB.SYS. For example:

```
:SEGMENTER

-SL SL.PUB.PRODUCT  <The PUB group of the PRODUCT account>
-USL USL.CC.DISC
-ADDSL FPROC'CC   <for OMNIQUIZ only>
-ADDSL TIMAGE11   <for BASIC only>
-ADDSL IMAGE'CC
-SL SL.PUB.SYS   <The system SL>
-USL USL.PUB.DISC
-ADDSL DISC'1
-ADDSL DISC'2
-ADDSL DISC'3
-ADDSL DISC'4
-ADDSL DISC'5
-ADDSL DISC'6
-ADDSL DISC'7
-ADDSL DISC_C
-ADDSL CCSCSEG0
-ADDSL CCSCSEG1
-ADDSL CCSCSEG2
-EXIT
```

Then, run your programs with LIB=P:

```
:RUN MINE.PROG.PRODUCT;LIB=P
```

Additional Notes

Explicit calls to the IMSAM intrinsics may be intermixed with implicit calls to IMSAM intrinsics made by programs under call conversion.

For example, if one of your existing programs is being run under call conversion, you can add calls to the Omnidex and IMSAM intrinsics to use these new retrieval capabilities. Here, you can still run the program under call conversion.

Another example is that of a vendor-supplied program for which you can write procedures to be executed while the program is running. With this type of program, you can use call conversion and add your own procedures with calls to the IMSAM intrinsics.

The call conversion library generates IMSAM intrinsic calls in response to certain IMAGE calls by the program, while the user-written procedures or programs call the IMSAM intrinsics directly, as needed.

When using call conversion, the IMSAM retrieval intrinsics DBIFIND and DBIGET also can be called indirectly. The call conversion library converts calls for DBFIND and DBGET to DBIFIND and DBIGET, passing all the parameters as is, except the *Status* array.

This means a Mode 7 DBGET is converted to a Mode 7 DBIGET, which is performed just like a DBGET Mode 7. This also means a Mode -104 DBGET (an illegal mode in IMAGE) is converted to a Mode -104 DBIGET, which allows you to do an IMSAM partial-key retrieval with a 4-byte key value.

In short, you can do IMSAM retrievals using IMAGE intrinsics by specifying IMSAM modes for DBGET and running the program under call conversion.

If calls to IMAGE intrinsics are mixed with calls to IMSAM intrinsics, then the program's error handling routines must be modified to call the appropriate intrinsics (DBERROR for IMAGE calls, DBIERROR for IMSAM calls). For example, if a program calls DBGET under call conversion, then DBERROR or DBEXPLAIN should be called if the call to DBGET fails. Conversely, if a program calls DBIGET, then DBIERROR or DBIEXPLAIN should be called if the call to DBIGET fails.

OMNIDEX Intrinsic Switch Stubs

Both versions of OMNIDEX can be called by either Native Mode or Compatibility Mode application programs. This is possible because of *switch stubs*, library routines that convert Compatibility Mode calls to Native Mode calls, and vice versa.

If you plan to call OMNIDEX Version 2.05 intrinsics from a Native Mode program, OMNIDEX includes a switch stub library file called XL.PUB.DISC. It is an executable library that functions as an interface between your Native Mode programs and the OMNIDEX Compatibility Mode intrinsic library (SL.PUB.DISC).

This means the switch stub library (XL.PUB.DISC) should be copied to the PUB group of the program account, along with the OMNIDEX intrinsic library (SL.PUB.DISC).

To use the OMNIDEX 2.05 switch stubs, use the **LIB=** parameter when running your Native Mode programs. MPE XL detects that you are running a Native Mode program and loads the XL switch stubs, which, in turn, calls the SL in the same group.

For example, you might have a Native Mode program that directly calls Omnidex or IMSAM intrinsics on an OMNIDEX-enhanced data base. The flow of control in such an instance is represented in **Figure 3 - 14** on the following page.

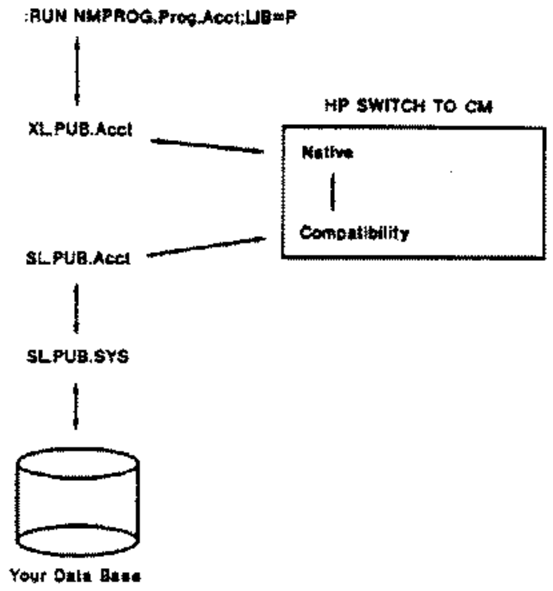


Figure 3 - 14: Calling OMNIDEX 2.05 from Native Mode.

Call Conversion Switch Stubs

The call conversion switch stubs are designed to enable your Native Mode programs to run against OMNIDEX-enhanced data bases without source code changes.

To run a program under call conversion in Version 2.05, you must use the call conversion switch stub (XL) and the intrinsic libraries (SL).

First, copy XL.CC.DISC and SL.CC.DISC into the group and account where the program resides.

Then, copy SL.PUB.DISC into the PUB group of the account where the program resides.

Use the **LIB=G** parameter when running your program.

```
:RUN nmprog.prog.acct;LIB=G
```

or

```
:RUN cmprog.prog.acct;LIB=G
```

Note: OMNIQUIZ Version 2.05 must be run with a Compatibility Mode, or Classic, version of QUIZ.

Writing Programs

Introduction

This section discusses in detail how to use the Omnidex and IMSAM intrinsic calls to perform the above functions.

- ❑ Open the data base;
- ❑ Lock parts of the data base at appropriate times;
- ❑ Update data set records and the corresponding index sets;
- ❑ Retrieve data set records using Omnidex and IMSAM capabilities;
- ❑ Retrieve data set records using standard IMAGE capabilities.

Opening the Data Base

A data base enhanced with Omnidex or IMSAM must be opened using the DBIOPEN intrinsic whenever the data base is accessed by a program.

DBIOPEN performs two functions that are crucial for subsequent access to the data base:

1. DBIOPEN calls DBOPEN to open the data base.
2. DBIOPEN allocates space for the Omnidex and IMSAM local control blocks (OLCB and ILCB). This space is allocated in virtual memory of the outer block area of the XL whenever a data base is opened.

Version 2.05

In Version 2.05 this space is allocated either in the program's stack when modes 1 through 8 or Control Option 100 are specified, or in an extra data segment if you use Control Option 800.

Omnidex and IMSAM intrinsics use these local control blocks during processing, so they must be created when the data base is opened.

Version 2.05

Fourth generation languages, which are run under call conversion, automatically use Control Option 800 for DBIOPEN. This is done to use an extra data segment for the local control blocks.

PASCAL requires the use of DBIOPEN Control Option 800 if any HELP commands are to be used; other third generation languages support it.

If you call DBOPEN instead of DBIOPEN, an error message, **Bad base ID, or data base not opened using DBIOPEN**, appears when you call an IMSAM or Omnidex intrinsic.

Note that DBIOPEN opens the data base a second time for internal use, but this second open uses negligible system resources and does not affect performance.

Locking Considerations

Before you add, delete or update a record using DBIPUT, DBIDELETE or DBIUPDATE with shared access to the data base, you must use the DBILOCK intrinsic to issue a lock.

This rule applies to any data set that contains an Omnidex keyword or IMSAM keyed field. You can specify a data base, data set or entry level lock by using DBILOCK modes 1 through 6.

DBILOCK automatically locks any index sets associated with the data set to index or update the Omnidex keywords and IMSAM keys for indexed fields in a record. The keywords and keys are updated appropriately after a DBIPUT, DBIDELETE or DBIUPDATE.

If you try to modify an entry in a set that has not been locked, and the set contains an Omnidex keyword or IMSAM keyed field, you receive an IMAGE error stating that DBPUT, DBDELETE or DBUPDATE has been called with no covering lock in effect.

If the set you are modifying contains Omnidex or IMSAM fields, but your application does not allow them to be modified, you can use DBILOCK with Mode Option 100 (IMAGE-only locking). This eliminates the unnecessary overhead of locking the index sets.

If the set you are modifying contains no Omnidex keyword fields or IMSAM key fields, you can use DBLOCK or DBILOCK with Mode Option 100 (IMAGE-only locking), as there are no index sets to lock. Still, it is better to use DBILOCK in case fields in the set are later enhanced with Omnidex or IMSAM.

Update Intrinsics

The three IMAGE intrinsics for updating information in an IMAGE data base are DBPUT, DBUPDATE and DBDELETE.

For Omnidex-enhanced or IMSAM-enhanced data sets, it is necessary to update the index sets as well as the data sets themselves.

The three IMSAM intrinsics that correspond to the standard IMAGE intrinsics and perform these updates are:

DBIPUT -	To add records.
DBIUPDATE -	To change data in existing records.
DBIDELETE -	To remove records.

These IMSAM intrinsics replace the corresponding IMAGE intrinsics. The syntax, which is provided in the **Intrinsics** chapter, is identical to the syntax of the IMAGE intrinsics.

There are three modes that provide three different methods of access for updating data sets enhanced with Omnidex and IMSAM. These modes are as follows:

- **Normal Mode**

  Normal mode is the most commonly used update mode. It is available for the DBIPUT, DBIUPDATE and DBIDELETE intrinsics.

  By calling any of these intrinsics with the *Mode* parameter set to **1**, the specified data record and the Omnidex index sets and IMSAM B-tree structures are updated immediately. This is the recommended mode for on-line applications.

- **IMAGE-Only Mode**

  IMAGE-only mode is used when you want to update data sets without updating their associated indexes. This mode is available for DBILOCK, DBIPUT, DBIUPDATE and DBIDELETE by adding 100 to the *Mode* parameter value when calling these intrinsics.

  This mode is typically used when converting data from an IMAGE data base to an Omnidex-enhanced or IMSAM-enhanced data base, or when performing large batch updates. After the data is added or updated in IMAGE-only mode, the indexes are repopulated using the ODXUTIL and DBIUTIL INDEX commands.

  If you wish to repopulate the IMSAM and Omnidex index sets in batch, use ODXPRO's JOB INDEX command. The ODXPRO prototyping and support program is discussed in the **Utilities** chapter of the **OMNIDEX Administrator's Guide.**

IMAGE-only mode is not recommended for on-line applications unless keyword retrieval and sorted-sequential access on data sets are not required immediately after updating. If you want to defer the updating of index sets, it is recommended that your administrator specify the Batch Indexing (BI) option during installation. See the **Keyword Field Options** section of the **Data Base Design** chapter and the **Omnidex Installation** section of the **Installation** Chapter of the **OMNIDEX Administrator's Guide** for more information.

❑ **Discrete Mode**

Discrete mode is used when you want to update the Omnidex and IMSAM index sets without updating the actual data set record. This mode is available for DBIPUT, DBIDELETE and DBILOCK only and is specified by adding 200 to the *Mode* parameter value when calling these intrinsics.

Discrete mode is also used with stand-alone B-tree structures and for Omnidex keywording of records by values that are not contained in a data record. See the **Stand-Alone IMSAM B-Trees** section of the **Data Base Design** chapter of the **OMNIDEX Administrator's Guide** for more information.

Omnidex Keyword Retrieval

To do keyword retrievals on an Omnidex data set, you must use the Omnidex keyword retrieval intrinsics, ODXFIND and ODXGET. The sequence of program calls is:

1. Call ODXFIND to locate the keywords in the indexes and create a list of qualifying Omnidex IDs for the records that contain them.
2. Call ODXGET to retrieve the Omnidex search item values (SIs) for the qualifying records.
3. Call DBGET or DBIGET to retrieve the actual record using the Omnidex SI value.

Note that ODXFIND qualifies IDs, but ODXGET retrieves SIs.

The following examples are common scenarios for data bases enhanced with Omnidex. The required intrinsic calls are described for each. Note that the last example describes how to view documents in the document management system after using keyword retrieval to qualify them.

On a Master Set

Keyword retrieval on a master set depends on whether there is only one group or field, or whether there are multiple groups or fields.

One Group or Field:

The program calls necessary for keyword retrieval by one group or field are:

1. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs (entries) qualified by the keywords specified.

2. **Call ODXGET (Mode 1)**

   Returns the Omnidex SI values (IMAGE search items that may be Omnidex IDs) for records that qualified.

   The number of SIs specified in the *SI-Count* parameter determines how many SIs are returned by each ODXGET. This call can be repeated to retrieve the next *SI-Count* Omnidex SIs.

3. **Call DBGET** or **DBIGET (Mode 7)**

   Retrieves the master record that qualified. This call can be repeated if more than one Omnidex SI was returned by ODXGET.

Note: Either DBGET or DBIGET (Mode 7) can be used.

Multiple Groups or Fields:

The program calls that are necessary for retrieval by multiple groups or fields are:

1. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs (entries) qualified by the keywords specified for the first group or field.

2. **Call ODXFIND (Mode 1)**

   Called once for each additional group or field, this ODXFIND returns the number of Omnidex IDs qualified by the previous ODXFIND, modified by the keywords specified in this call.

The keyword list must start with an asterisk (*). The asterisk (*) is the SAMELIST operator. It loads the most recent keyword list for further Boolean operations. SAMELIST AND (*,) reduces the qualification to the IDs that are common to both the current ID list and the list of IDs qualified by the new keywords. SAMELIST OR (*+) increases the qualification of IDs qualified by the new keywords, or in both. SAMELIST NOT (*-) reduces the qualification to the IDs that are in the current list, but that are not in the list of IDs qualified by the new keywords.

Note that the SAMELIST operations are performed after all other Boolean operations specified in the keyword list have been completed.

3. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs qualified by the previous calls to ODXFIND, plus this call for the third group or field. Repeat this call for all desired keyword groups or fields using * as the first byte within the Keyword list.

4. **Call ODXGET (Mode 1)**

   Returns the Omnidex SI values (the IMAGE search items) for records that qualified. The number specified in the *SI-Count* parameter determines how many SIs are returned by each ODXGET. This call can be repeated to return the next *si-count* Omnidex SIs.

5. **Call DBGET or DBIGET (Mode 7)**

   Retrieves the master record that qualified. This call is repeated for each Omnidex SI that is returned by ODXGET.

On a Detail Set

Like retrieval on master sets, keyword retrieval on detail sets depends on whether there is one group or field, or whether there are multiple groups of fields. It also depends on whether the detail set is a detail-record indexed (DR) set or contains record-specific (RS) fields.

One Group or Field:

The program calls necessary for keyword retrieval on one group or field are:

1. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs (record complexes) qualified by the specified keywords.

2. **Call ODXGET (Mode 1)**

   Returns the Omnidex SI values (the IMAGE search items) for the record complexes that qualified.

Because Omnidex returns the search item values, you must also read the chain defined by each search item to find the specific detail record you require. The number in the *SI-Count* parameter determines how many SIs are returned by each ODXGET. This call can be repeated to return the next n Omnidex SIs.

3. **Call DBFIND** or **DBIFIND (Mode 1)**

   Uses the Omnidex SI value to set up a pointer for a chained read to the detail data set. This call can be repeated for each Omnidex SI value.

4. **Call DBGET** or **DBIGET (Mode 5)**

   Retrieves the detail record. This call can be repeated to read all entries in a chain.

Multiple Groups or Fields:

The program calls for keyword retrieval on a detail using multiple groups or fields are:

1. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs (record complexes) qualified by the keywords specified for the first group or field.

2. **Call ODXFIND (Mode 1)**

   Called once for each additional group or field, this ODXFIND returns the number of Omnidex IDs qualified by the previous ODXFIND, modified by the keywords specified in this call.

   The keyword list must start with an asterisk (*). The asterisk (*) is the SAMELIST operator, described on the previous page. It loads the most recent keyword list for further Boolean operations.

3. **Call ODXGET (Mode 1)**

   Returns the Omnidex SI values (the IMAGE search items) for the record complexes that qualified.

   Because Omnidex returns the IMAGE search items, you must also read the chain defined by that search item (chain head) to find the specific detail record you require. The number in the *SI-Count* parameter determines how many SIs are returned by each ODXGET. This call can be repeated to return the next *n* Omnidex SIs.

4. **Call DBFIND or DBIFIND (Mode 1)**

   Uses the Omnidex SI value to set up a pointer for a chained read to the detail data set. This call can be repeated for each Omnidex SI value.

5. **Call DBGET** or **DBIGET (Mode 5)**

   Retrieves the detail record. This call can be repeated to read all the entries in a chain.

<u>**DR Detail Set:**</u>

Keyword retrieval on a DR detail set requires the following program calls:

1. **Call ODXFIND (Mode 1)**

   Returns the number of detail records qualified by the specified keywords.

2. **Call ODXGET (Mode 1)**

   Returns the record number for the first detail record. This call can be repeated for each detail record.

3. **Call DBGET** or **DBIGET (Mode 4)**

   Retrieves the first detail record. This call can be repeated for each detail record.

<u>**RS Detail Set:**</u>

Keyword retrieval on a detail set with RS fields requires the following program calls:

1. **Call ODXFIND (Mode 1)**

   Returns the number of detail records qualified by the specified keywords.

2. **Call ODXGET (Mode 11)**

   Returns the record number for the first detail record. This call can be repeated for each detail record.

3. **Call DBGET** or **DBIGET (Mode 4)**

   Retrieves the first detail record. This call can be repeated for each detail record.

Other Omnidex Retrievals

Omnidex also can retrieve only a qualifying count, retrieve only keywords or retrieve external documents and view them on-line.

Qualifying Count:

To retrieve only a qualifying count (not the records), the program call is:

1. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs qualified by the keywords specified.

Only Keywords:

To retrieve only keywords, the program calls are:

1. **Call ODXFIND (Mode 10 or 11)**

   Returns the number of keywords (not the number of Omnidex IDs) that qualify.

2. **Call ODXGETWORD (Mode 1 or 2)**

   Returns the first keyword that qualified. This call can be repeated to retrieve each qualifying keyword.

Qualify Documents:

To qualify documents and view them on-line, the program calls are:

1. **Call ODXFIND (Mode 1)**

   Returns the number of Omnidex IDs qualified by the keywords specified.

   This ODXFIND (or ODXFINDs) must be performed on the data set that contains the catalog entries.

2. **Call ODXGET**

   Returns the Omnidex SI values for the catalog entries that qualified.

3. **Call DBGET or DBIGET (Mode 7)**

   Retrieves the catalog entry.

4. **Call ODXVIEW**

   Displays the document that corresponds to a catalog entry in a data base.

For more information about catalog entries, see the **OMNIDEX Document Management Supplement.**

<u>**Multifind retrievals:**</u>

Multifind From Memory

To do multifind retrievals from memory, an Omnidex key field in the target domain must contain values common to the Omnidex search item of the source domain. For example, if the Omnidex SI of the source domain is PRODUCT-NO, you can retrieve records in another domain on a PRODUCT-NO Omnidex keyword field in another domain.

To multifind from memory:

1. **Call ODXFIND (mode 1)** on a key in the set of the source domain.

Returns the number of Omnidex IDs qualified by the keywords specified. Repeat this call for all desired keyword groups or fields, using * as the first character of the keyword list after the first call.

2. **Call ODXFIND (Mode 1)** on a key in a set in the target domain.

   The field parameter should contain the name of the field that contains values common to the Omnidex search item in the source domain. The keywords parameter should contain only an ampersand (&). This causes Omnidex to use the search items qualified in the first domain as keyword arguments linked by an OR (+) operation on the target key. You can further qualify the list by performing additional ODXFINDs on other fields in the target set using an asterisk (*) as the first character in the keywords parameter.

Multifind from a file

To do multifind retrievals across databases or using fields that are not Omnidex search items, you must first create a file containing the qualified items from the source domain. This file can then be used as input for an Omnidex retrieval in the target domain.

1. **Call ODXFIND (Mode 1)** on a key in the set of the source domain.

   Returns the number of Omnidex IDs qualified by the keywords specified. Repeat this call for all desired keyword groups or fields, using * as the first character of the keyword list after the first call.

2. **Call ODXTRANSFER (mode 1 or 2 using mode option 100 or 200).**

   If you wish to transfer values from a field that is also the Omnidex search item, call ODXTRANSFER mode 201 (for a new file) or 202 (to append to a file). Since Omnidex search items are stored internally, this process is extremely fast.

   If you want to transfer values from a field other than the Omnidex search item, call ODXTRANSFER mode 101 (for a new file) or 102 (to append to a file). The the CONTROL parameter of ODXTRANSFER should contain the data set and item name of the field being transferred.

   Because Omnidex must retrieve each qualified master record, this process will be much slower than ODXTRANSFER using mode option 200.

3. **Call ODXFIND (mode 1)** on a key in a set of the target domain.

   The field parameter should contain the name of the field that contains values common to the Omnidex search item in the source domain. The keywords parameter should contain an ampersand (&), followed by the file name specified in the call to ODXTRANSFER.

   Omnidex will use values contained in the file as keyword arguments linked by an OR (+) operation on the target key. You can further qualify the list by performing additional ODXFINDs on other fields in the target set using an asterisk (*) as the first character in the keywords parameter.

IMSAM Keyed Access

To retrieve a record using IMSAM keyed access (which provides partial-key and sorted-sequential retrieval), you must use the IMSAM intrinsics instead of the standard IMAGE intrinsics.

The following examples describe possible scenarios for IMSAM partial-key and sorted-sequential retrievals and the intrinsic calls for each. Note that although partial-key and sorted-sequential retrievals are described separately, they are usually combined.

In most IMSAM retrievals, DBIGET is the only intrinsic used.

The only exception to this is using an IMSAM key that is also the search item in a master set. If you want to read the detail chain for a master record, you must first call DBIFIND to set up chain pointers to the detail(s).

For IMSAM retrievals using DBIGET, it does not matter whether the IMSAM key is for a master set or a detail set.

The following examples show standard IMSAM retrievals and IMSAM retrievals that set up the chain head.

Note that Mode 100 is =, 200 is >, 300 is >=, 400 is <, and 500 is <=.

Also note that for each partial-key retrieval using DBIGET or DBIFIND, the length of the partial-key value used in the search must be added to the mode. This is described after the examples.

Partial-Key Retrieval

To do a partial-key retrieval on a master set or detail set that contains an IMSAM key or composite key, the program calls are:

1. **Call DBIGET (Mode 100, 200, 300, 400 or 500 plus the number of words or bytes of the partial key value. A negative mode indicates bytes, a positive mode indicates words.)**

   Returns a record to the specified buffer area of the calling program. This record is the first record in sequence that qualifies based on the partial-key value and the mode used.

To do sorted-sequential retrievals on a master or detail set that contains an IMSAM key or composite key, the program calls are:

1. **Call DBIGET (Mode 100, 200, 300, 400 or 500)**

   Retrieves the first record in sequence and sets up a pointer to subsequent records.

2. **Call DBIGET (Mode 90 or 91)**

   Retrieves the next record in sequence. This call can be repeated to retrieve all the subsequent records in sequence by key.

Note that no preceding DBIFIND is required to retrieve a detail record via an IMSAM key within a detail set. This is because IMSAM uses the record number to index the IMSAM key values for a detail set.

Values are stored in an IMSAM B-tree index as follows:

- For an IMSAM key that is also an IMAGE search item in a master data set, the key stored in the IMSAM B-tree is the IMAGE search item value.

- For an IMSAM-only key (an IMSAM key that is not an IMAGE search item) in a master data set, the key stored in the IMSAM B-tree includes the IMSAM key value plus the IMAGE search item value.

  For example, a master data set might have an X2 field called STATE specified as an IMSAM key and an X4 IMAGE search item called ACCT.

  If the values in these fields for a particular record are ACCT = 123 and STATE = CO, the full internal key value in the B-tree would be **CO123**.

- For an IMSAM key in a detail data set, the key stored in the IMSAM B-tree includes the IMSAM key value plus the IMAGE record number.

  For example, if STATE was an IMSAM key and ACCT was the IMAGE search item in a detail data set, the key in the IMSAM B-tree would be **CO** plus the record number. If the record number was 978, the full key would be **CO (978)** where (978) is a double-word integer value.

  For more information about record numbers, see the DBGET intrinsic, Mode 4, in the IMAGE manual.

You must determine programmatically when to terminate the index sequential reads. DBIGET does not return an exceptional condition on a sequential read unless the beginning (i.e., the first key) or end (i.e., the last key) of a file is reached.

Setting Up Chain Heads

To do a partial-key retrieval on a detail using a search item that is an IMSAM key in the associated master, the program calls are:

1. **Call DBIFIND (Mode 100, 200, 300, 400 or 500 plus the number of words or bytes of the partial value)**

   Points to the first SI in sequence and calls DBFIND on that SI.

2. **Call DBGET** or **DBIGET (Mode 5)**

   Retrieves the first detail record using a standard IMAGE chained read. This call can be repeated for all the records you want to retrieve from the chain.

The program calls to perform sorted-sequential retrievals on a detail set using a search item that is an IMSAM key in the associated master are as follows:

1. **Call DBIFIND (Mode 100, 200, 300, 400 or 500)**

   Points to the first detail in a chain for the first qualifying key value.

2. **Call DBIGET** or **DBGET (Mode 5)**

   Retrieves the first detail record using a standard IMAGE chained read. This call can be repeated for all the records in the chain that you want to retrieve.

3. **Call DBIFIND (Mode 90 or 91)**

   Retrieves the next SI in sequence and calls DBFIND on that SI.

4. **Call DBIGET** or **DBGET (Mode 5)**

   Retrieves the first detail record using a standard IMAGE chained read. This call can be repeated for all the records in the chain that you want to retrieve.

Using Partial Keys

When you do IMSAM partial-key retrievals, you must specify how many words or bytes are to be used for comparison during the IMSAM search. To do this by adding the length of the partial value specified by the user to the base mode value of 100, 200, 300, 400 or 500. A positive value specifies the length in words, while a negative value specifies the length in bytes. For example, **104** means the partial key is 4 words long, while a mode of **-104** means the partial key is 4 bytes long.

For example, the following values might exist for an IMSAM key called DATE:

871215
880101
880202
890107
890108
890109

If the partial value, **88**, was used as the argument, the following date values would be retrieved with the various IMSAM modes:

Relop	Mode	Retrieved	Reason
=	100	not found	A *
=	101	880101	B **
>	200	880101	A
>	201	890107	B
>=	300	880101	A
>=	301	880101	B
<	400	871215	A
<	401	871215	B
<=	500	871215	A
<=	501	880101	B

* Reason A = **88** ____ used for comparison

** Reason B = **88** used for comparison

If you do an IMSAM retrieval without specifying the length to compare for a partial value, an exact comparison is made on the full length of the key.

Discrete Mode Retrievals

Discrete mode is used when you want to retrieve an IMSAM key without the corresponding record.

For example, you might have a transaction data set in a general-ledger data base with an IMSAM key called TRANS-KEY, which is a composite key comprised of the transaction DATE and AMOUNT. To find the transaction amounts for a particular date or month, you would not need to retrieve the data records themselves. All you would need are the key values.

For this retrieval, you could use **8701** as a partial value for the Argument in a discrete mode DBIGET on the TRANS-KEY composite key.

This would return all the transactions for January 1987 in sorted-sequential order.

The calls would be:

1. **Call DBIGET (Mode 1102)**

   Retrieves the first key in sequence and sets up a pointer to subsequent keys.

2. **Call DBIGET (Mode 1090)**

   Retrieves the next key in sequence. This can be repeated for retrievals of the subsequent keys in sequence.

The TRANS-KEY values retrieved for this composite key might be:

870101	100.01
870102	1874.56
870104	533.94
870110	12.50
870115	19820.34

.

.

Discrete mode retrievals improve performance tremendously when only the key itself is required. This is because the B-tree set used to index an IMSAM key contains many keys in each physical record. For the above example, 3 I/Os would be required to retrieve the first 25 keys and 1 I/O would be required for every 25 keys thereafter.

Conversely, if normal mode was used instead of discrete mode for the above retrieval, it would require 3 I/Os for the first key and 1 I/O for each key thereafter. This amounts to 103 I/Os to retrieve 100 transactions using normal mode, versus only 6 I/Os for discrete mode. Obviously, discrete mode would be much more efficient.

Note: You must use discrete mode for all IMSAM DBIGET retrievals when using the OMNIDEX interface with QUICK and TRANSACT/3000. The IMAGE search item passed back with the IMSAM key can then be used to retrieve the actual records.

The only exception to this is that normal mode DBIGETs must be performed on detail sets in QUICK. See the **OMNIDEX Guide for Interfacing with PowerHouse** and the **OMNIDEX Guide for Interfacing with RAPID** for more information about performing DBIGETs.

IMAGE Retrievals

Any standard IMAGE intrinsic can be used to retrieve data from an OMNIDEX-enhanced data base. Any IMSAM intrinsic can use a standard IMAGE mode, which causes the call to be passed directly to the IMAGE intrinsic. For example, calling DBIGET with a Mode 7 is the same retrieval as a DBGET Mode 7.

Although you can use both IMAGE and IMSAM calls in the same program, it is recommended that you use IMSAM calls instead of IMAGE calls throughout your programs. This provides consistency and guarantees that the locking is adequate.

Interfacing with 3GLs

Introduction

The advanced retrieval capabilities of Omnidex and IMSAM can be called easily through third generation languages (3GLs) like BASIC, COBOL, FORTRAN, PASCAL, C, SPL and RPG.

The previous section of this chapter outlined the sequence of the intrinsic calls you would use within an application program. The **Intrinsics** chapter discusses the syntax for these intrinsic calls.

This section describes the parameters you use when you call the Omnidex and IMSAM intrinsics. It also provides examples of COBOL programs so you can see how these calls are used.

If you would like to see examples of OMNIDEX applications written in PASCAL, FORTRAN, BASIC or RPG, refer to the **OMNIDEX Language Sampler**. Source files of programs which call OMNIDEX intrinsics through BASIC, COBOL, FORTRAN and PASCAL are also available in the DEMO group of your DISC account.

Parameters and Definitions

This section lists and defines the parameters needed for programs that call the Omnidex and IMSAM intrinsics.

The parameters and their sizes are:

Parameter	COBOL
Argument	Key value format
Base	X (4) or greater
Buffer	Size varies
Control	S9 (4) COMP occurs 5
Dset	X (32)
*Filename**	X (36) or less
*Info**	Size varies
Item	X (16) or less
*Keywords**	X (100), for example
Mode	S9 (4) COMP
*Plabels**	S9 (4) COMP occurs 10
*Sl-count**	S9 (4) COMP
*Sl-list**	S9 (9) COMP for example
Status	S9 (4) COMP occurs 21

Table 3 - 1: Intrinsic parameter definitions

Parameters that are marked by an asterisk (*) are used by Omnidex intrinsics but not by IMSAM intrinsics.

Most of these parameters for the IMSAM and Omnidex intrinsics are identical to the corresponding parameters for IMAGE intrinsics. The only differences are the size of *Status*, the *Dset* parameter for DBIGET and an additional parameter for DBIEXPLAIN.

The parameters are defined as follows. See the **Intrinsics** chapter for the syntax of each intrinsic.

Argument	The *Argument* parameter for an IMAGE-only mode DBIFIND or DBIGET defines the value of a search item for retrieval. For an IMSAM DBIFIND or DBIGET, the *Argument* parameter defines the value (or partial value) of an IMSAM key to search for. For an Omnidex ODXFIND, the *Keywords* parameter is used instead of the Argument parameter.
Base	The *Base* parameter is defined the same as in IMAGE.
Buffer	The *Buffer* parameter is defined the same as in IMAGE.
Control	The *Control* parameter is used in the ODXVIEW and ODXPRINT intrinsics to provide information about the file being viewed or printed.
Dset	For all the intrinsics except DBIGET, *Dset* is defined the same as in IMAGE. The *Dset* parameter in the DBIGET intrinsic is used to specify which data set and which IMSAM key within the data set to use for IMSAM access. The first 16 characters (16 bytes) are reserved for the data set name and the last 16 are used for the name or number of the IMSAM key. Both must be terminated by a blank () or a semicolon (;).
Filename	The *Filename* parameter is used in the ODXVIEW intrinsic.
Info	The *Info* parameter is used in the ODXINFO intrinsic.
Item	For IMAGE, the *Item* parameter defines the search item name or number. For IMSAM, it defines an IMSAM key name or number. For Omnidex, it defines a keyword field name or number.
Keywords	The *Keywords* parameter is used in the ODXFIND intrinsic. It passes the list of keywords, separated by Boolean operators, specified for retrieval. Note that this must always be an ASCII buffer, even for retrievals on binary fields.

Mode	The *Mode* parameter is defined the same as in IMAGE. However, additional values can be passed through the *Mode* to obtain IMSAM retrievals. Note that for random mode DBIFIND and DBIGET retrievals, the length of the partial key is added to the *Mode*.
Plabels	The *Plabels* parameter is used in the ODXVIEW and ODXPRINT intrinsics.
SI-Count	The *SI-Count* parameter is used in the ODXGET intrinsic.
SI-List	The *SI-List* parameter is used in the ODXGET intrinsic. It must be defined as the same type as the Omnidex search item and must be large enough to hold the number of SIs specified by *SI-count*.
Status	The *Status* parameter returns information about the procedure call. It is 21 16-bit-words (42 bytes) for a data base enhanced with Omnidex or IMSAM. Words 1 through 21 are defined on the following page.

Word	Use	COBOL
1	IMAGE condition word	S9(4) COMP
2	IMAGE entry length	S9(4) COMP
3-4	IMAGE current record number	S9(9) COMP
5-6	IMAGE chain length	S9(9) COMP
7-8	IMAGE backward pointer	S9(9) COMP
9-10	IMAGE forward pointer	S9(9) COMP
11	Omnidex/IMSAM status word	S9(4) COMP
12-13	Omnidex ID count	S9(9) COMP
14-15	reserved for Omnidex	S9(9) COMP
16-17	used by ODXFIND	S9(9) COMP
18-19	used by ODXFIND	S9(9) COMP
20-21	used by ODXFIND	S9(9) COMP

Table 3 - 2: Status words in COBOL and FORTRAN

Status words 1-10 are reserved for IMAGE. They are not used by the Omnidex or IMSAM intrinsics. The only exception is *Status* word 1, which is set to 888 for Omnidex errors or to 999 for IMSAM errors.

If *Status* word 1 is 888 or 999, *Status* word 11 contains the IMSAM or Omnidex error condition.

For information about how words 12 through 21 are used, see the description of each Omnidex intrinsic in the **Intrinsics** chapter.

COBOL Program Examples

This section provides examples of COBOL code for each IMSAM and Omnidex retrieval intrinsic.

Note that the following examples are only partial listings of COBOL programs. Use the sample programs in the DEMO group of the DISC account for more complete examples of interfacing COBOL with Omnidex and IMSAM.

Also note that the SPL notation for comments (<< comment >>), which is occasionally used in these examples, is not allowed in COBOL.

DBIFIND

Use DBIFIND only when an IMSAM key is also an IMAGE search item in a master, and you require subsequent retrieval of detail records.

```
PROGRAM-ID.  programname.
   .
   .
01 RECORD-BUFFER.
   .
   .
01 DSET.
   03  DATA-SET PIC X(16).
   03  IMSAM-key     PIC X(16).
   .
   .
01 ITEM          PIC X(16).
   .
   .
PROCEDURE DIVISION.
   .
   .
   .
100-FIND-RECORD.
   .
   .
   MOVE "imsamdetailset;" TO DATA-SET.
   MOVE "imsamkey;" TO ITEM.
   << Note that the key must be an IMAGE search item >>
   MOVE -305 TO MODE.
   << Assume a 5-byte partial key >>
   MOVE "partialkey" TO ARGUMENT.
   CALL "DBIFIND" USING BASE, DSET, MODE, STATUS,
   ITEM, ARGUMENT.
```

```
      IF IMAGE-STATUS NOT EQUAL 0 THEN
         IF IMS-ODX-STAT =210= 210
         OR IMS-ODX-STAT =211= 211
         OR IMS-ODX-STAT =217 THEN
         DISPLAY "Key value not found"
      ELSE
         PERFORM ERROR-ROUTINE
         GO TO END-OF-PROGRAM.


   200-GET-RECORD.

      MOVE 5 TO MODE.
      CALL "DBIGET" USING BASE, DSET, MODE, STATUS,
      LIST, BUFFER, ARGUMENT.
      IF IMAGE-STATUS = 15 THEN
         GO TO 300-FIND-NEXT.
      IF IMAGE-STATUS NOT EQUAL 0 THEN
         PERFORM ERROR-ROUTINE
         GO TO 100-FIND-RECORD.

      DISPLAY RECORD-BUFFER.
      GO TO 200-GET-RECORD.


   300-FIND-NEXT.

      MOVE 90 TO MODE.
      CALL "DBIFIND" USING BASE, DSET, MODE, STATUS,
         ITEM, ARGUMENT.
      IF IMAGE-STATUS NOT EQUAL 0 THEN
         IF IMS-ODX-STAT = 211 THEN
            DISPLAY "End of file"
            GO TO END-OF-PROGRAM
         ELSE
            PERFORM ERROR-ROUTINE
         GO TO END-OF-PROGRAM.
   GO TO 200-GET-RECORD.
      .
      .
      .
 ERROR-ROUTINE.
 CALL "DBIEXPLAIN" USING STATUS, PARM.
      << Note additional parameter! >>
```

DBIGET

Use DBIGET for all types of IMSAM keys in both master and detail data sets.

Refer to the file COBIMSS.DEMO.DISC for a sample program that shows normal mode and discrete mode calls to DBIGET.

```
PROGRAM-ID.  programname.
   .
   .
01 RECORD-BUFFER.
   .
   .
01 DSET.
   03  DATA-SET PIC X(16).
   03  IMSAM-KEY     PIC X(16).
   .
   .
01 ITEM         PIC X(16).
   .
   .
PROCEDURE DIVISION.
   .
   .
100-GET-RECORD.
   .
   .
   MOVE "imsamdataset;" TO DATA-SET.
   MOVE "imsamkey;" TO IMSAM-KEY.
   MOVE -305 TO MODE.<< Assuming 5 byte partialkey >>
   MOVE "partialkey" TO ARGUMENT.
   CALL "DBIGET" USING BASE, DSET, MODE, STATUS,
      LIST, BUFFER, ARGUMENT.
   IF IMAGE-STATUS NOT EQUAL 0 THEN
      IF IMS-ODX-STAT = 310
      OR IMS-ODX-STAT = 311
      OR IMS-ODX-STAT = 317 THEN
         DISPLAY "Key value not found"
         GO TO 100-GET-RECORD
      ELSE
         PERFORM ERROR-ROUTINE
         GO TO END-OF-PROGRAM.
   .
   .
   .
ERROR-ROUTINE.
   CALL "DBIEXPLAIN" USING STATUS, PARM.
```

ODXFIND

Refer to the file COBODXS.DEMO.DISC for another program example that shows Omnidex keyword retrievals using ODXFIND (and ODXGET).

```
PROGRAM-ID.  programname.
   .
   .
100-BEGIN.
   .
   .
   MOVE 1 TO MODE.
   MOVE "dsetname" TO DSET.
   MOVE "fieldname" TO ITEM.
   MOVE "keyword1,keyword2,..." TO KEYWORDS.
      << Terminate list with blank or semicolon >>
   CALL "ODXFIND" USING BASE, DSET, MODE, STATUS,
      ITEM, KEYWORDS.
   IF IMAGE-STATUS NOT EQUAL 0 THEN
      IF IMS-ODX-STAT = 217 THEN
         DISPLAY "Keyword not found"
         GO TO 100-BEGIN
      ELSE
         PERFORM ERROR-ROUTINE
   ELSE
      DISPLAY REC-COUNT.


200-GET-RECORDS.
   .
   .
   .
ERROR-ROUTINE.
   CALL "DBIEXPLAIN" USING STATUS, PARM.
```

ODXGET

ODXGET requires a preceding call to ODXFIND with a mode of 1 or 2.

Refer to the file COBODXS.DEMO.DISC for another program example that shows Omnidex keyword retrievals using ODXFIND (and ODXGET).

```
PROGRAM-ID.  programname.
   .
   .
01  SI-LIST.
   03  SI-LIST-ID  PIC S9(9) COMP.
   .
   .
200-GET-RECORDS.
```

```
     MOVE 1 TO SI-COUNT.
     MOVE 1 TO MODE.
     CALL "ODXGET" USING BASE, MODE, STATUS, SI-LIST,
        SI-COUNT.
     IF IMAGE-STATUS NOT EQUAL 0 THEN
        IF IMS-ODX-STAT = 311 THEN
           DISPLAY "End of qualifying records"
           GO TO 100-BEGIN
        ELSE
           PERFORM ERROR-ROUTINE
              GO TO 100-BEGIN
     ELSE
        PERFORM 300-GET-RECORDS.
  300-GET-RECORDS.

     MOVE SI-LIST-ID TO ARGUMENT.
     MOVE 7 TO MODE.
     MOVE "datasetname" TO DSET.
     CALL "DBGET" USING BASE, DSET, MODE, STATUS,
        LIST, BUFFER, ARGUMENT.
        << Note that DBGET is used for the IMAGE mode 7 Get >>
     IF IMAGE-STATUS NOT EQUAL 0 THEN
     .
     .
     ELSE
        PERFORM 400-DISPLAY-REC.
     .
     .
  ERROR-ROUTINE.
     CALL "DBIEXPLAIN" USING STATUS, PARM.
```

ODXGETWORD ODXGETWORD requires a preceding call to ODXFIND with a mode of 10 or 11.

```
  PROGRAM-ID.  programname.
     .
     .
  200-GET-KEYWORDS.
     .
     .
     MOVE 2 TO MODE.
     CALL "ODXGETWORD" USING BASE, MODE, STATUS,
        TARGET.
     IF IMAGE-STATUS NOT EQUAL 0 THEN
        PERFORM ERROR-ROUTINE
        GO TO 200-GET-KEYWORDS.
```

```
      DISPLAY TARGET, STATUS-WORDS-12-13.
      GO TO 200-GET-KEYWORDS.
      .
      .
      .
   ERROR-ROUTINE.
      CALL "DBIEXPLAIN" USING STATUS, PARM.
```

ODXTRANSFER

Refer to the file COBODXS.DEMO.DISC for an example.

```
   PROGRAM-ID.  programname.
      .
      .
   03  FILENAME       PIC X(06)  VALUE "ODXIDS".
      .
      .
   TRANSFER-ENTRIES.
      CALL "ODXTRANSFER" USING DBN-PATH1, MODE1, DB-STATUS,
         FILENAME, ODXTR-OPT.
      IF IMAGE-STATUS NOT = 0 THEN
         PERFORM ERROR-DISPLAY
         MOVE YES TO DONE.
         GO TO TRANSFER-ENTRY-EXIT.

      PERFORM CREATE-REPORT THRU CREATE-REPORT-EXIT.
   TRANSFER-ENTRY-EXIT.
   EXIT.
```

The report that uses the search item values from the ODXTRANSFER is as follows:

```
   CREATE-REPORT.
      OPEN INPUT ODXID.
      OPEN OUTPUT REPORT-TR.
      MOVE SPACES TO FILLER1.
      MOVE SPACES TO FILLER2.
      MOVE SPACES TO FILLER3.
      READ ODXID RECORD INTO OMNSIEX-SI AT END MOVE
         YES TO DONE.
      PERFORM PRINT-RECORD THRU PRINT-RECORD-EXIT
         UNTIL DONE = YES.
      CLOSE ODXID.
      CLOSE REPORT-TR.
   CREATE-REPORT-EXIT.
      EXIT.
```

Interfacing with 4GLs

The advanced retrieval capabilities of Omnidex and IMSAM can be implemented easily with fourth generation languages (4GLs) such as:

- Speedex™ from Infocentre
- DATA Express™ from IMACS Corporation.
- Insight™ from Unison Software Inc.
- Synergist™ from Gateway Systems Corporation
- Visimage™ from Ares Corporation
- QUICK™ from Cognos Corporation's Powerhouse system
- TRANSACT/3000™ from Hewlett-Packard's Rapid system
- Protos™ from Protos Software Corporation

Infocentre's Speedex has integrated the Omnidex intrinsics, so Speedware products automatically interface with OMNIDEX and require no additional programming.

Unison Software's Insight and Gateway's Synergist have embedded the IMSAM intrinsics, so they automatically interface with IMSAM and require no additional programming. Synergist also can call the IMSAM intrinsics from a personal computer.

Other 4GLs can also be used with OMNIDEX. Documentation on QUICK, TRANSACT/3000 or PROTOS can be obtained from DISC. Call your DISC sales representative or DISC Technical Services for additional documentation.

Troubleshooting

Troubleshooting Procedures

Before calling our phone-in support line, there are several troubleshooting procedures you should perform. They are:

1. If there is an abort or a similar problem, perform a PSCREEN to obtain a printed copy of the error message or problem situation. To do this, enter:

   ```
   :RUN PSCREEN.UTIL.DISC
   ```

   PSCREEN is a contributed library program that takes a snapshot of the terminal screen and sends it to the line printer.

   If the error occurs while running DATADEX, write down the error. Then, try to duplicate the error, and enter **PSCREEN** at the **Command?** prompt. Note that PSCREEN is not supported in dumb mode.

2. Check the version numbers of the software to make sure they are all compatible. To do this, first run DBIUTIL by entering:

   ```
   :RUN DBIUTIL.PUB.DISC;LIB=P
   ```

   After you access a data base, data set, and IMSAM key, enter the VERSION command to display the software version number and the version number under which the OMNIDEX IMS was installed on the data base.

3. If you have moved the OMNIDEX Intrinsics Library XL.PUB.DISC to another account where your programs reside, you should find the version of the intrinsics by using the LINKEDIT Utility. To do this, run the LINKEDIT program at an MEOP system prompt:

   ```
   :LINKEDIT
   ```

   Next check the version number of the intrinsics:

   ```
   LinkEd >XL XL.PUB.youracct
   LinkEd >LISTXL ;MODULE=DBIOPEN.ASM
   LinkEd >EXIT
   ```

   A number of entry points scroll past. There is an entry point called **OMNIDEX'*#####***. The five numbers (*#####*) represent the version of the intrinsics in the account.

Version 2.05

If you have moved the OMNIDEX Intrinsics Library SL.PUB.DISC to another account where your programs reside, you should find the version of the intrinsics by using the MPE SEGMENTER:

```
SEGMENTER
-SL SL.PUB.youracct
-LISTSL DISC'4
-EXIT
```

A number of entry points scroll past. There is an entry point called **OMNIDEX'#####**. The five numbers (#####) represent the version of the intrinsics in the account.

If this version number does not match the one in SL.PUB.DISC (the version number shown by the DBIUTIL VERSION command), then you may have a problem with incompatible versions of the software.

4. You can use ODXMGR to generate an IMAGE schema. Make sure that no items or sets have 0-class security.

```
:RUN ODXMGR.PUB.DISC,OMNIDEX
Data base name: dbname.group
.
.
ODXMGR: SCHEMA
```

5. If the problem occurred with an application program, you should use DATADEX to verify that the same problem can be reproduced using the DATADEX commands.

   Because the DATADEX commands call the OMNIDEX IMS intrinsics, they are very helpful for debugging. The DATADEX commands and their corresponding intrinsics are listed in Table A - 1, on the following page.

6. For more advanced troubleshooting techniques, you can run DBIUTIL again and use the VERIFY and LIST commands to inspect the keys in the B-tree. Refer to the **Utilities** chapter of the **OMNIDEX Administrator's Guide** for more detailed information about the commands.

DATADEX Command	Corresponding Intrinsics
ADD	DBIPUT
COPY	DBIPUT
DELETE	DBIDELETE
FIND	DBIGET (*Mode* depends on relational operator); DBIFIND for IMSAM keys in master sets; DBIGET Mode 5 for IMAGE-only keys in detail sets
GET	DBIGET (any IMAGE or IMSAM mode, including IMAGE serial read and IMSAM discrete modes)
MODIFY	DBIGET Mode 1, DBIUPDATE
MO	DBIGET Mode 7, DBIUPDATE for masters; DBIFIND Mode 1, DBIGET Mode 5, DBIUPDATE for details
NEXT	DBIGET Mode 90 for masters; DBIFIND Mode 90, DBIGET Mode 5 for details
NO	ODXGET Mode 1, DBIGET Mode 7 for masters; ODXGET Mode 1, DBIFIND Mode 1, DBIGET Mode 5 for details
OMNIDEX	ODXFIND Mode 1 or 2 for standard retrieval; ODXGET if List is not configured; ODXGET Mode 1, DBIGET Mode 7 for masters; ODXGET Mode 1, DBIFIND Mode 1, DBIGET Mode 5 for details if List is configured
OMNIDEX (keyword only retrieval)	ODXFIND Mode 10 or 1
List? Y	ODXGETWORD Mode 2
PREVIOUS	DBIGET Mode 91 for masters; DBIFIND Mode 91, DBIGET Mode 5 for details
PO	ODXGET Mode 2, DBIGET 7 for masters; ODXGET Mode 2, DBIFIND Mode 1, DBIGET Mode 5 for details
REREAD	DBIGET Mode 1
SAMEKEY	DBIGET Mode 5 or 6
VIEW	ODXVIEW
DBINFO	DBIINFO or ODXINFO
Prompt for data base, password, and mode	DBIOPEN
QUIT or EXIT from data set prompt	DBICLOSE

Table A - 1: DATADEX Commands and their Corresponding Intrinsics

Also note that a DBILOCK is performed before a DBIPUT, DBIDELETE, or DBIUPDATE on a data base opened without exclusive access, and a DBIUNLOCK is performed afterwards.

Common Programming Errors

Some common programming errors to check for are:

1. The status array must be 21 16-bit words (42 bytes).
2. DBIPUT requires an @ item list, or the equivalent, for a data set that contains Omnidex keyword fields, unless the set uses a transparent Omnidex ID (TR option) or is a DR indexed detail set.
3. The mode for a DBIGET must include the number of characters for the key length, unless the full key is entered.
4. After performing a DBIGET on a key value and a DBIGET Mode 90 or 91 (GET SEQUENTIAL) to retrieve the records, you must programmatically perform a check to find when the key value changes. DBIGET does not automatically know when to stop the retrieval.
5. Any calls to VOPENFORMF should execute prior to the first call to DBIOPEN.
6. QUICK requires discrete mode DBIGETs on a master set and normal mode DBIGETs on a detail set.
7. When using ITOQ to create a Cognos source schema, make sure to either run ITOQ before installing the OMNIDEX IMS on a data base, or to remove definitions for the added OMNIDEX sets and items from your Cognos source schema file before running QDD.
8. If QUICK under CALL Conversion is not updating the data base, you should check the QSCHEMA to make sure that it specifies the same data base and group as the ODX'INTRINSICS file.

   Be sure that the CALL Conversion SLs are installed in the same group where QUICK resides.
9. When using FORTRAN 77, modes, status arrays, and parms must be declared as **INTEGER*2.**

Version 2.05

10. When using the NEW function in Pascal, DBIOPEN Mode 800 must be used.

11. When defining RECORD definitions in PASCAL/XL or STRUCTures in C/XL, make sure that the individual members are packed in the manner that you expect. A member that commonly causes problems is the OMNIDEX ID count, which is a 32-bit integer, aligned on a 16-bit boundary.

Miscellaneous Errors

1. Do not abort jobs that update.

DBIFIND and DBIGET Errors

When you program with the IMSAM intrinsics DBIFIND and DBIGET, you sometimes encounter the exceptional conditions **Beginning of File**, **End of File**, or **Key not found.**

The error numbers for these exceptional conditions are:

DBIFIND	DBIGET	Message	Abbreviation
210	310	End of file	EOF
211	311	Beginning of file	BOF
217	317	Key not found	KNF

Table A - 2: IMSAM exceptional conditions

When you reach the **End of File**, the record pointer is at the last record. When you reach the **Beginning of File**, the record pointer is at the first record.

The conditions under which you may see one of these error messages depend on the retrieval mode. A table of the retrieval modes and actions is provided on the following page. Note the following assumptions for the examples:

- Only two key values, **1** and **3**, are in the data set. **2** does not exist.
- Retrievals on the lowest possible value in collating sequence are referred to by the COBOL term, *Low Values.*
- Retrievals on the highest possible value in collating sequence are referred to by the COBOL term, *High Values.*

Examples of DBIFIND or DBIGET retrievals are:

Mode	**Key Value**	**Error**	**Action**
100 (=)	Low Values	210 or 310 (BOF)*	Use Mode 92 to FIND or GET first record value of 1.
100 (=)	High Values	211 or 311 (EOF)*	Use Mode 92 to FIND or GET last record value of 3.
100 (=)	2	217 or 317 (KNF)*	Use different key value.
200 (>)	Low Values		None. Record 1 found or retrieved.
200 (>)	High Values	211 or 311 (EOF)*	Use Mode 92 to FIND or GET last record value of 3.
200 (>)	2		None. Record 3 found or retrieved.
300 (>=)	Low Values		None. Record 1 found or retrieved.
300 (>=)	High Values	211 or 311 (EOF)*	Use Mode 92 to FIND or GET last record value of 3.
300 (>=)	2		None. Record 3 found or retrieved.
400 (<)	Low Values	210 or 310 (BOF)*	Use Mode 92 to FIND or GET first record value of 1.
400 (<)	High Values		None. Record 3 found or retrieved.
400 (<)	2		None. Record 1 found or retrieved.
500 (<=)	Low Values	210 or 310 (BOF)*	Use Mode 92 to GET 1st record value of 1.
500 (<=)	High Values		None. Record 3 found or retrieved.
500 (<=)	2		None. Record 1 found or retrieved.

Table A - 3: Examples of DBIFIND or DBIGET Retrievals

* BOF stands for Beginning of File
EOF stands for End of File
KNF stands for Key Not Found

Changes that Require Re-installation

The following table summarizes some of the possible changes and the required procedures:

Changes	Required procedures*
DBUNLOAD/DBLOAD	1, 2, 3
RELOAD DR detail sets or detail sets that have IMSAM keys	2
Add or remove values from the excluded words list	3
Change keyword options	3
Structural change	1, 2, 3

Table A - 4: Changes to an Omnidex Data Base that Require Maintenance

Note: A structural change involves adding or deleting an item, set or path in a way that affects Omnidex or IMSAM as discussed in the **Installation** chapter of the **OMNIDEX Administrator's Guide.**

***Key to procedures**

1	Re-install Omnidex and IMSAM using DBINSTAL.
2	Re-index IMSAM B-tree(s) using DBIUTIL INDEX operation.
3	Re-index Omnidex indexes using ODXUTIL INDEX operation.

Changes can be made using Hewlett-Packard's DBChange or Adager's Itemadd, Itemdel, Setadd, Setdel, Fieldadd, Fielddel, Pathadd or Pathdel. Also, Adager's Itemchng and Setname may affect Omnidex set names.

The procedures that do not require any changes are:

- ❑ Change a set capacity.
- ❑ Change a primary path (unless RS fields are present).
- ❑ Copy a data base.
- ❑ Rename a data base.

System Resource Considerations

Version 2.05

Stack Overflow

A stack overflow abort occurs when there is not enough room in the data stack for the program's global and local variables. Ways to avoid this problem are:

1. MAXDATA should be 30,000 or more. You can try running the program with the ;NOCB parameter to place the PCBX in an extra data segment. This saves 400-500 words of stack, and may be enough to eliminate the stack overflow problem.
2. Cut down on unnecessary program variables and reduce sizes of tables or arrays. You can also split a large program into smaller subroutines to move variables from global storage, which is always loaded, to local storage, which in loaded only when needed.
3. Limit the amount of indexing that you add to the data base. This reduces the size of the OLCB and ILCB (Omnidex and IMSAM Local Control Blocks), and thereby reduces the amount of stack they require. You should also use the default of 512 when prompted for the tree block size in DBINSTAL.
4. Use Mode 2 for ODXFIND instead of Mode 1 for programs that use VIEW, in order to return the stack space after the CALL. You should also use the VPLUS FAST forms file option via the FORMSPEC utility so that only the records containing information needed at run-time are kept in the FORMSPEC file.
5. Use the $CONTROL DYNAMIC option for compiling COBOL subroutines.
6. Use input/output files instead of procedures for the sort for programs that use SORT 3000 and perform Omnidex or IMSAM calls in the output procedure. For COBOL programs, you can use the command $CONTROL SORTSPACE = *n* at the top of the program and specify a value greater than the default of 1000. This reserves a greater amount of stack space for use by procedures other than the sort. These steps are recommended because SORT 3000 does not return the stack space it uses; therefore, there may not be enough stack for the Omnidex or IMSAM intrinsic calls after a sort.
7. Do not use DBIOPEN Mode 800 or CALL Conversion, if possible, when using SORT. DBIOPEN with mode option 800 moves the OLCB and ILCB into the stack for each intrinsic call, and they may not fit in the small amount of stack left after a sort. DBIOPEN Mode 1 or 5 places the local control blocks in stack before the sort, requiring no additional stack after the sort.

8. If you are using QUICK, there are several things you can do to reduce stack space usage:

   a. Avoid using temporary variables or literals (a character string enclosed in quotation marks). You should also avoid the use of ERROR, SEVERE, INFO, or WARNING with literals. A better method is to set ERROR equal to *ODX'MESSAGE*, where *ODX'MESSAGE* is a defined variable.

   b. Set up the QKGO file with a small to medium number of PROC-CODE pages, and minimize the parameters for rollback and memory buffer size.

   c. Minimize the number of levels of QUICK screens and menus. If you need multiple levels of screens, we recommend that you set up two QKGO files and divide the levels into two parts. When you run QUICK, it should access the first QKGO file, which goes to the first screen or menu. At the middle level, file equate QKGO to the second QKGO file, and run QUICK again to access the next lower screen. This sets up a second area, and you can process-handle between the two parts.

9. Use DBIOPEN mode option 800 to avoid stack overflows when using the NEW function in Pascal or when opening multiple data bases.

10. Make sure you have placed the CALL Conversion SL (SL.CC.DISC) and the Intrinsics SL (SL.PUB.DISC) (or the corresponding USLs) in the correct places. If you place them both in the same group, a stack overflow occurs due to an endless loop of one SL routine calling another.

11. Reduce the size of the excluded words list.

Configuring OMNIDEX for Your Site

The DBINSTAL CONFIG command enables you to accommodate limitations in your system's stack and disk space. The DBINSTAL CONFIG command is discussed below.

Note: Although you are prompted for a data base name when you run DBINSTAL, these configurations apply to USL.PUB.DIS and SL.PUB.DISC. Therefore, you may need to install SL.PUB.DISC to other accounts after configuration changes have been made.

You must have exclusive access to the SL and USL in PUB.DISC to use any of the CONFIG options.

First, run DBINSTAL, and enter the name of the data base:

```
:RUN DBINSTAL.PUB.DISC
.
.
Data base name: dbname.group.acct
```

You are prompted for an operation. Enter C, for **CONFIG**:

```
Enter a command at the prompt or ? for help.

DBINSTAL: C
```

The CONFIG options are displayed, and you are asked to specify one of them:

```
CONFIG options are:
      TTable
      Exit

Config option?
```

See the Loading a Translation Table section at the end of this chapter for more information about the TTABLE option.

Version 2.05

Version 2.05 offers several additional CONFIG options that can be used to configure the way OMNIDEX makes use of stack space:

```
 CONFIG options are:
       Changelink
       Dsegsize
       Filesize
       Headroom
       TTable
       Exit

 Config option?
```

Each of these options is discussed below.

Changelink

Specify the Changelink option by entering C at the **Config option?** prompt. This option is used to modify the location of the Omnidex Local Control Block (OLCB) and IMSAM Local Control Block (ILCB) pointer in the DL-DB area of stack. This is useful for preventing stack contention between your application programs and the Local Control Block pointer. The pointer is established to identify the location of the OLCB and ILCB.

The figure to the right shows some possible locations for the Local Control Block pointer. The configurable locations are marked with an asterisk.

Also shown are some of the words of the DL-DB area that are reserved for use by the operating system and various programming languages. You may want to consult your programming language's manual for more complete information about its use of the DL-DB area.

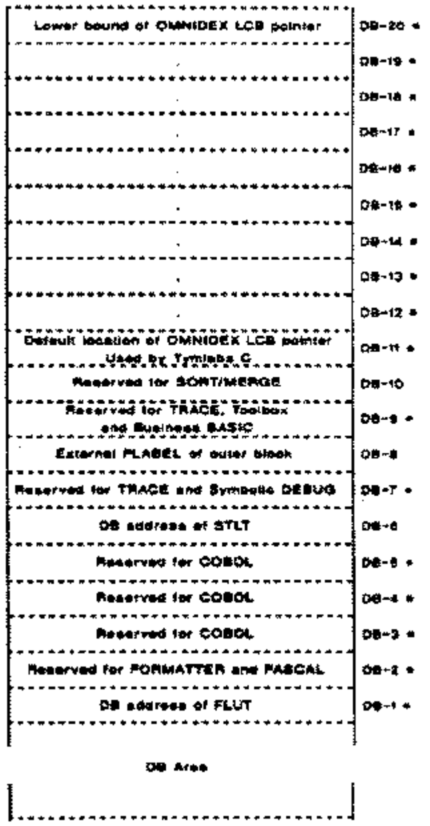
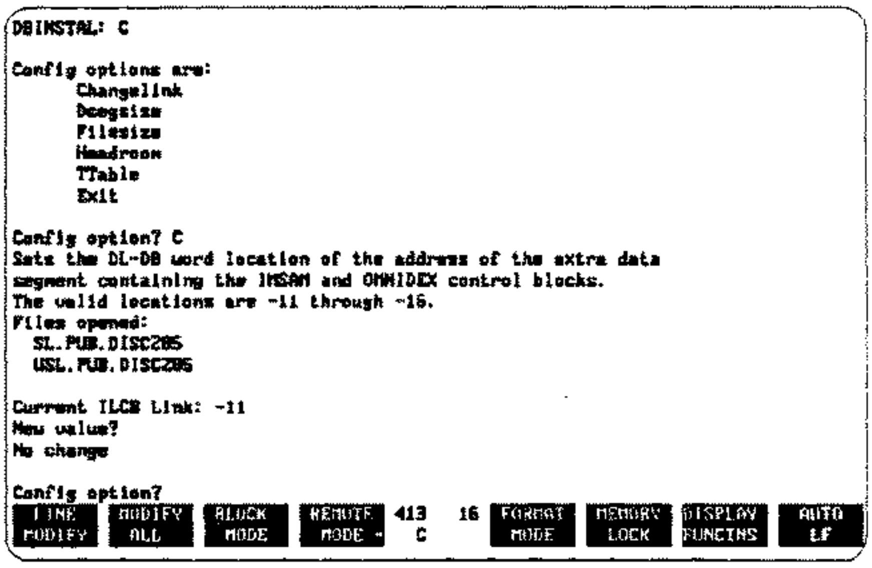


Figure A - 1: Configurable locations for the DBIOPEN Mode 800 local control block pointer.

When you enter C, the option's function is described, and the SL and USL in the PUB group of the DISC account are serially read to determine the current setting. The current location of the Local Control Blocks is displayed, and you are prompted to enter a new value. Valid locations for the ponter are displayed as follows:

```
The valid locations are -1 through -5, -7, -9, and -11
through -20.
```

You can enter a new value, or press ↵ to leave the location of the Local Control Blocks unchanged. The prompts are shown in Figure A - 2 below.



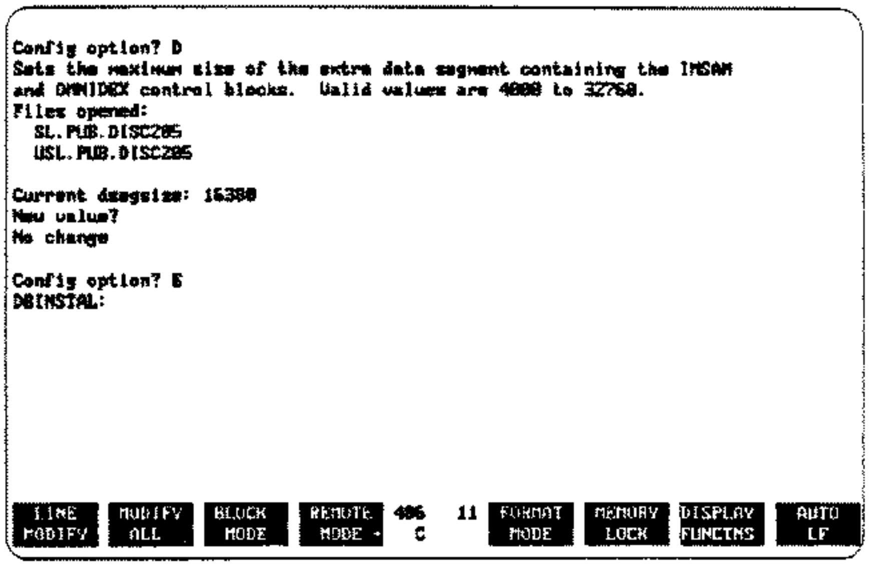
```
DBINSTAL: C

Config options are:
       Changelink
       Dsegsize
       Filesize
       Headroom
       TTable
       Exit

Config option? C
Sets the DL-DB word location of the address of the extra data
segment containing the IMSAM and OMNIDEX control blocks.
The valid locations are -11 through -16.
Files opened:
  SL.PUB.DISC285
  USL.PUB.DISC285

Current ILCB Link: -11
New value?
No change

Config option?
```

Figure A - 2: Configuring the location of the local control blocks through DBINSTAL

Dsegsize

Specify the Dsegsize option by entering **D** at the **Config option?** prompt. This option is used to modify the maximum size of the extra data segment created for DBIOPEN Mode 800. The size of this data segment determines the number of OLCBs and ILCBs that can be placed in it. This in turn determines the number of DBIOPENs allowed per process. The default size is 16,380 words and the acceptable values range from 4000 to 32,760 words.

When you enter **D**, the option's function is described, and the SL and USL in the PUB group of the DISC account are serially read to determine the current setting. The current maximum data segment size is displayed, and you are prompted to enter a new value (between 4000 and 16380 words). You can enter a new value, or press ↵ to leave the data segment size unchanged. The prompts are shown in Figure A - 3, below:



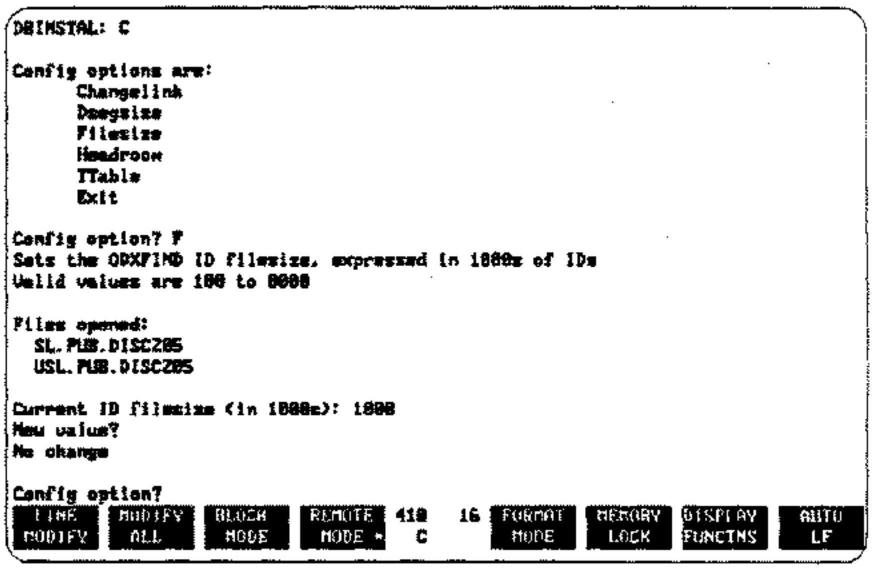
```
Config option? D
Sets the maximum size of the extra data segment containing the IMSAM
and OMNIDEX control blocks.  Valid values are 4000 to 32760.
Files opened:
  SL.PUB.DISC205
  USL.PUB.DISC205

Current dsegsize: 16380
New value?
No change

Config option? E
DBINSTAL:
```

Figure A - 3: Configuring the maximum size of extra data segments through DBINSTAL

Filesize

Specify the Filesize option by entering **F** at the **Config option?** prompt. This option is used to modify the size of the session-temporary ODXFIND ID files used to hold Omnidex IDs during an Omnidex search. This is useful for sizing these files to accommodate a scarcity or surplus of disk space.

When you enter **F**, the option's function is described, and the SL and USL in the PUB group of the DISC account are serially read to determine the current setting. The current ID file size is displayed (in thousands (1000s) of IDs), and you are prompted to enter a new value (between 100 and 8000). You can enter a new value, or press ↵ to leave the ID file size unchanged. The prompts are shown in Figure A - 4 below.



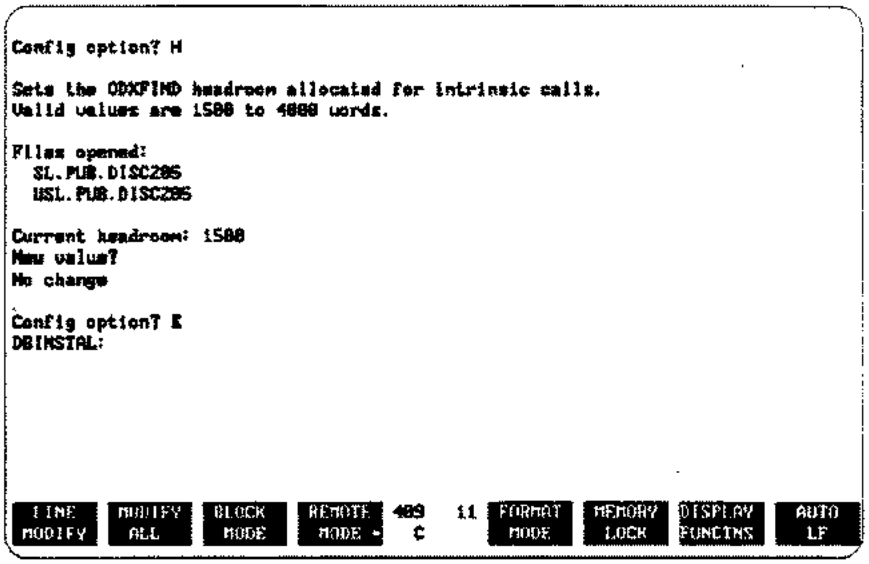
```
DBINSTAL: C

Config options are:
       Changelink
       Dsegsize
       Filesize
       Headroom
       TTable
       Exit

Config option? F
Sets the ODXFIND ID filesize, expressed in 1000s of IDs
Valid values are 100 to 8000

Files opened:
  SL.PUB.DISC205
  USL.PUB.DISC205

Current ID filesize (in 1000s): 1000
New value?
No change

Config option?
```

Figure A - 4: Configuring the size of odxfind id files through DBINSTAL

Headroom

Specify the Headroom option by entering **H** at the **Config option?** prompt. This option is used to modify the amount of ODXFIND headroom allocated for other intrinsic calls within ODXFIND. This is most useful for 'NS' and 'DS' sites where the standard file system intrinsics require two to five times more stack. Default is 1500 words and the acceptable values range from 1500 to 4000.

When you enter **H**, the option's function is described, and the SL and USL in the PUB group of the DISC account are serially read to determine the current setting. The current headroom is displayed, and you are prompted to enter a new value (between 1500 and 4000 words). You can enter a new value, or press ↵ to leave the ODXFIND headroom size unchanged. The prompts are shown in Figure A - 5, below.



```
Config option? H

Sets the ODXFIND headroom allocated for intrinsic calls.
Valid values are 1500 to 4000 words.

Files opened:
  SL.PUB.DISC205
  USL.PUB.DISC205

Current headroom: 1500
New value?
No change

Config option? E
DBINSTAL:
```

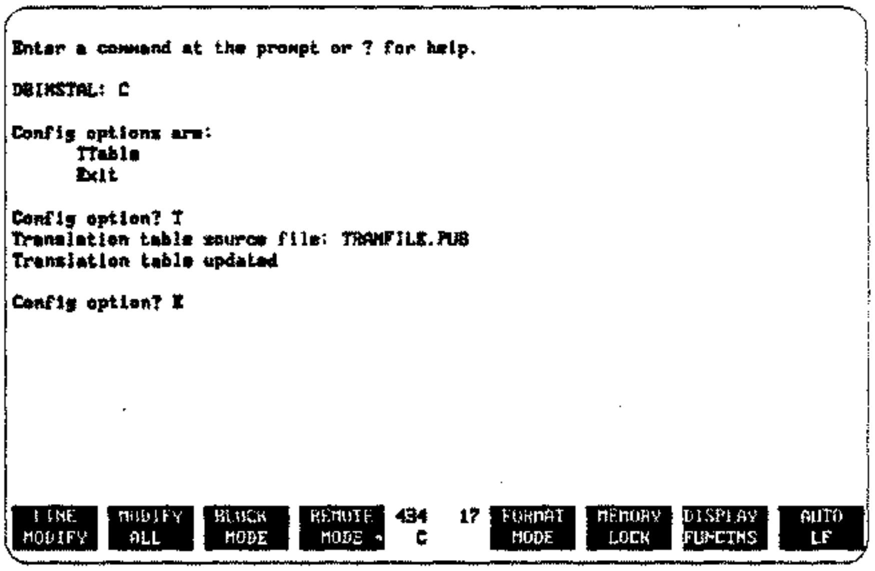
Figure A - 5: Configuring the ODXFIND headroom through DBINSTAL

Loading a Translation Table

If you are loading a translation table to handle the translation of Kana8, Arabic8, Greek8 or Turkish8, use the TTable option. Remember, the translation of Roman8 is handled automatically and does not require a translation table.

To load the translation table, enter **TTable** at the **Config option?** prompt. DBINSTAL asks for the **Translation table source file:**. Enter the name of the source file. When DBINSTAL is finished loading the translation table, a message, **Translation table updated**, is displayed and the **Config option?** prompt appears again. The entire process should resemble the prompts shown in Figure A - 6, below.

For information about creating a translation table, see the **8-Bit Character Sets** section of the **Data Base Design** chapter in the **OMNIDEX Administrator's Guide.**



```
Enter a command at the prompt or ? for help.

DBINSTAL: C

Config options are:
       TTable
       Exit

Config option? T
Translation table source file: TRANFILE.PUB
Translation table updated

Config option? E
```

Figure A - 6: Loading a translation table

Index