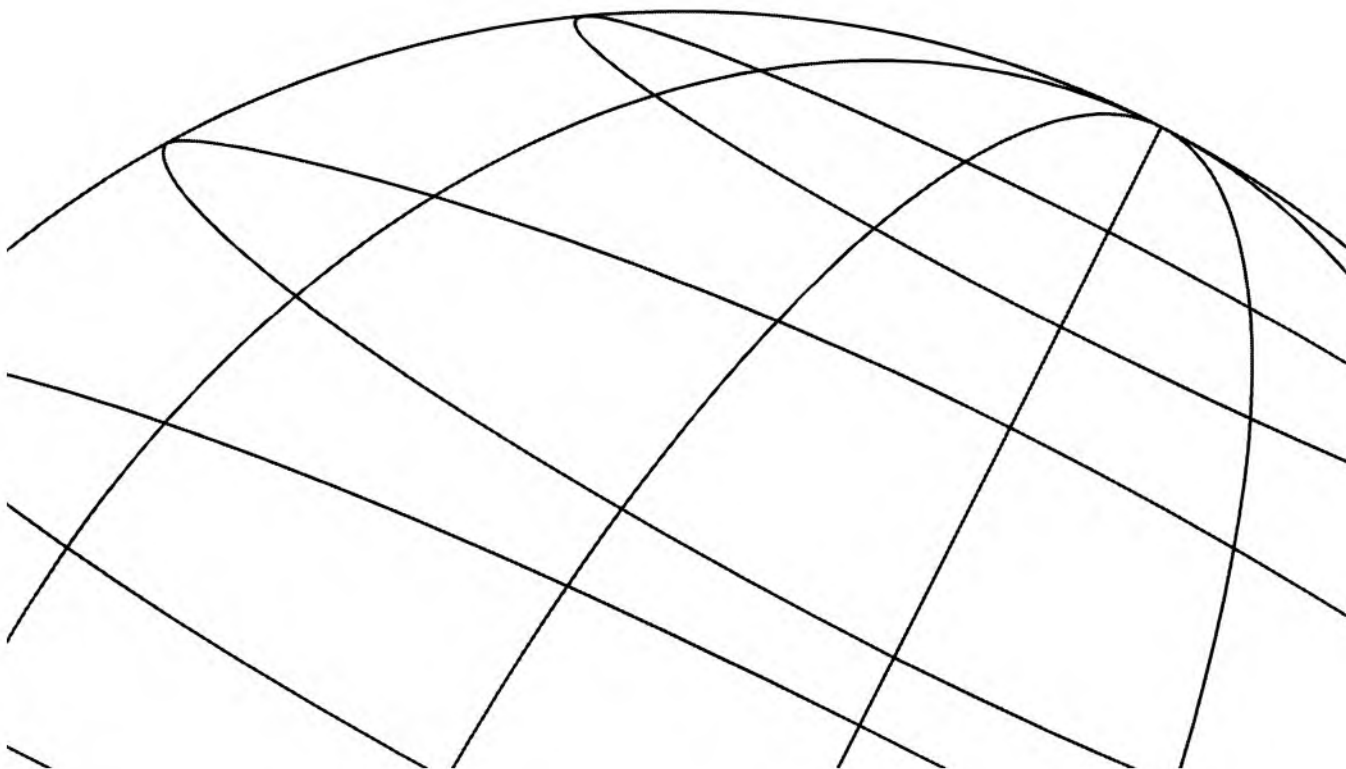


OMNIDEX

Business
Solutions for
the 90's

***OMNIDEX ImagePlus SDK
Administrator's Guide for HP MPE/iX
Version 3.4***



Dynamic Information Systems Corporation, Boulder 80301

Version 3.4

Fourth edition: June 1996

The information contained in this document is subject to change without notice. Dynamic Information Systems Corporation makes no warranty of any kind with respect to the sufficiency or fitness of this information for a particular purpose. Dynamic Information Systems Corporation is not liable for errors contained herein or incidental or consequential damages in connection with the furnishings, performance, or use of this material.

©Copyright 1996 by Dynamic Information Systems Corporation. All rights reserved. Permission is granted to reprint this document, but not for profit.

DISC

Dynamic Information Systems Corporation

5733 Central Avenue

Boulder, Colorado 80301

(303) 444-4000

DISC Europe

25-29 High Street Leatherhead

Surrey, England KT228AB

+44 1372-362777



PRINTED ON RECYCLED PAPER

Table of Contents

Using this Guide	xi
Overview	xi
Conventions.....	xi
How this guide is organized.....	xii
Technical support.....	xiii
Acknowledgments	xiv

Chapter 1: Introduction

Welcome to Version 3 of OMNIDEX.....	1-2
What are keyword keys?.....	1-2
Searching with keyword keys	1-3
What are sorted keys?.....	1-6
Searching with sorted keys	1-7
Why Use OMNIDEX?.....	1-8
Speed of access to data	1-8
Retrieval flexibility.....	1-9
Ease of installation and maintenance	1-10
Faster application development.....	1-10
The Standard Interface to Third Party Indexing (TPI)	1-10
The OMNIDEX Intrinsic Interface	1-11
Reliability	1-11
Implementing OMNIDEX.....	1-12

Components of OMNIDEX	1-13
The OMNIDEX utilities	1-13
OmniUtil	1-13
DataView	1-14
DBMGR	1-15
Intrinsic libraries	1-15
The OMNIDEX intrinsic library	1-15
The Call Conversion library	1-16
The Standard Interface to Third Party Indexing (TPI)	1-17
Switch stub intrinsic library	1-17
Why wait any longer for your data?	1-18

Chapter 2: OMNIDEX Installation

Before you begin	2-2
Installing OMNIDEX Keys	2-3
Selecting a table (data set)	2-3
Keying fields	2-3
Creating composite keys	2-4
Adding key options	2-7
Keyword key options	2-8
Sorted key options	2-14
Keyword and sorted key options	2-14
Finishing the installation for a table	2-16
Correcting mistakes	2-16
Adding keys to a table	2-16
Removing keys from a table	2-17
Changing keys for a table	2-17
Activating the Installation	2-19
Real-time activation	2-20
Saving an installation	2-20
Installation and indexing jobs	2-21
OMNIDEX index structures	2-21

Excluding Words from Indexing	2-22
Creating an excluded words file	2-22
Loading the excluded words list	2-22
Customizing the Translation of 8-bit Characters	2-24
Creating a translation table	2-24
Loading the translation table	2-26
Loading the Indexes	2-28
Indexing in real time	2-29
Indexing jobs	2-30
Index buffer size	2-30
Keyword index sorting	2-31
Verifying and Testing the Installation	2-32
Finding and listing records	2-32
Changing the Installation	2-34
Reindexing changes	2-35

Chapter 3: Utilities

OmniUtil	3-2
Program operation	3-2
Using the menus	3-3
Using dialog boxes	3-3
Prompt windows	3-4
Using the function keys	3-4
Running in trace mode	3-5
OmniUtil menus	3-6
The Main Menu	3-6
Index Installation and Maintenance	3-8
Reindexing options	3-11
Show Information	3-13
Configuration Options	3-17

DataView	3-21
Program operation.....	3-21
Running DataView.....	3-21
Using DataView windows.....	3-23
Using the function keys.....	3-23
The Main Menu.....	3-24
Finding records.....	3-26
Maneuvering through prompts.....	3-26
Selecting records using keyword keys.....	3-27
Selecting records using sorted keys.....	3-31
Listing and viewing records.....	3-32
DBMGR	3-34
Program operation.....	3-34
Timing estimates for DBMGR operations.....	3-35
CAP[acity].....	3-35
RELoad.....	3-35
ERAsE.....	3-36
COPy.....	3-37
DBMGR and security.....	3-37
Logging.....	3-38
Intrinsic Level Recovery.....	3-38
Third Party Indexing.....	3-38
Released databases.....	3-39
Break/abort.....	3-39
Concurrent processes.....	3-39
Structural problems.....	3-40
IMAGE and TurboIMAGE.....	3-40
Job mode processing.....	3-41
Sample job streams.....	3-41
INFO string processing.....	3-42
DBMGR commands.....	3-43
Listing DBMGR commands.....	3-44
The BASE command.....	3-45
The CAPacity command.....	3-45
The COPy command.....	3-48
The DELetE command.....	3-49
The ERAse command.....	3-50
The Exit command.....	3-51

The FOrM command	3-51
The Help command	3-52
The PRImary command.....	3-52
The QUIT command	3-53
The RELoad command	3-53
The REName command.....	3-55
The SCHema command.....	3-56
Contributed OMNIDEX Utilities	3-58

Chapter 4: Topics

OMNIDEX Retrievals.....	4-2
Keyword searches	4-2
Qualifying records	4-3
Searching across keys	4-3
Searches across tables	4-5
Null selections	4-6
Retrieving records.....	4-6
Keyword arguments	4-6
Keyword search operations	4-12
Sorted key retrieval capabilities	4-16
Relational operations	4-17
Ranges	4-19
Sorted sequential reads	4-19
The OMNIDEX Intrinsic Interface	4-20
Programming considerations	4-20
Additional retrieval features	4-21
The Standard Interface to Third Party Indexing	4-24
Enabling OMNIDEX searches	4-25
Additional retrieval features	4-25
Determining Retrieval Requirements	4-28
Noting keys for installation	4-28
Ordered data vs. free-form data.....	4-29
Which fields make good keys?	4-30
Fields used in KSAM files	4-30
Key fields.....	4-31
Textual and descriptive fields	4-31

Hierarchical fields	4-32
Code fields	4-32
Date fields	4-33
Reports as indicators	4-33
OMNIDEX retrievals or serial reads?	4-34
Reports to convert to online.....	4-35
Retrieval possibilities.....	4-36
Combining keyword arguments	4-37
Searching several keys.....	4-37
Searching across data sets	4-38
Searching parts and combinations of fields	4-38
Data retrieved	4-41
Designing composite keys	4-43
Composite keyword keys.....	4-44
Composite sorted keys	4-46
Comparison of keyword and sorted keys.....	4-49
Updating OMNIDEX Data	4-50
Real time updates.....	4-50
Automatic indexing through TPI.....	4-51
Automatic indexing through Call Conversion	4-51
Batch Updates.....	4-52
Using the Batch Indexing option.....	4-53
Disabling Third Party Indexing for a database	4-54
Performing IMAGE-only updates.....	4-55
Linking Details to Masters	4-56
OMNIDEX masters	4-56
OMNIDEX details	4-56
OMNIDEX domains.....	4-57
SI domains.....	4-57
Searching across data sets	4-58
Record complexes	4-58
Split Retrievals.....	4-60
Compressing record specific retrievals	4-61
Detail Record (DR) domains.....	4-62
Multifind	4-63
When tables share a search item (SI)	4-63

When tables share common data values	4-64
Requirements.....	4-64
Executing a Multifind operation	4-65
Multifind from memory	4-65
Multifind from an external file	4-66
Maintenance	4-68
Reinstalling OMNIDEX keys.....	4-69
Updating an existing OMNIDEX installation.....	4-70
Changing key installation	4-72
Reinstalling as a maintenance measure.....	4-72
Installation jobs	4-76
Reindexing.....	4-78
When is it necessary?.....	4-78
Reindexing jobs	4-80
Maintenance and Third Party Indexing (TPI).....	4-81
Determining whether TPI Is enabled.....	4-81
Turning TPI on or off.....	4-83
Index updates and the Transaction Manager	4-85
Recovery from a system failure.....	4-87
Maintaining access to OMNIDEX indexes.....	4-88
Maintaining OMNIDEX index file capacities	4-88
How OMNIDEX determines index file capacities	4-89
Checking index file capacities	4-89
Situations requiring increased index file sizes	4-90
Changing the index file capacities	4-91
Deinstalling OMNIDEX keys.....	4-93
Removing OMNIDEX with OmniUtil	4-93
Other Performance Considerations	4-96
Estimating disk space requirements.....	4-96
Keyword index files and unload files.....	4-96
Sorted key index files.....	4-97
IMAGE-only batch updates	4-98
Disabling Third Party Indexing for maintenance operations.....	4-98
Tuning indexing performance.....	4-99
Number of keywords to be indexed	4-100
Use of memory	4-101
Re-specifying excluded words	4-101

Streamlining OMNIDEX Databases	4-102
Possible modifications.....	4-102
Eliminating automatic masters.....	4-103
Eliminating sorted paths.....	4-103
Eliminating KSAM files.....	4-104
Changing detail sets into manual masters.....	4-104
Adding an integer search item (SI).....	4-104
Logical design changes.....	4-105
Changing details to masters.....	4-105
Hierarchies greater than two levels.....	4-105
Changing database structure.....	4-107
Step 1: Revising and compiling the schema.....	4-108
Step 2: Creating the modified database.....	4-108
Step 3: Transferring data.....	4-109

Appendix A: Data Type Considerations

Integer fields (Types I, J or K).....	A-1
Zoned fields (Type Z).....	A-2
Packed fields (Type P).....	A-2
Real fields (Type R and E).....	A-3
Character fields (Types U or X).....	A-3
TurboIMAGE field arrays (compound Items).....	A-3
Alternate data types.....	A-4
ASK and PowerHouse date fields.....	A-4
Data type discrepancies.....	A-5

Appendix B: OMNIDEX Parsing Rules

Parsing during indexing.....	B-1
Keyword key values.....	B-1
Parsing data for sorted keys.....	B-5
Parsing during keyword searches.....	B-5
Preventing argument parsing.....	B-7
How OMNIDEX translates 8-bit characters.....	B-8

Appendix C: Installing OMNIDEX Procedure Libraries

The OMNIDEX intrinsic library	C-2
The TurboIMAGE/iX intrinsics	C-3
The Call Conversion library	C-3
The Intrinsic Switch Stub library	C-4
Referencing and placing libraries	C-4
Referencing XLs.....	C-4
Using the Switch Stub SL	(SL.PUB.DISC) C-6
Using alternate OMNIDEX intrinsic libraries.....	C-8

Appendix D: OMNIDEX Version 3 Limits

Opens per process.....	D-1
Database maximums.....	D-1
Number of index files.....	D-1
Number of OMNIDEX domains.....	D-1
Number of keyword keys	D-2
Number of excluded words	D-2
Number of tables (data sets) that contain sorted keys.....	D-2
Number of sorted keys	D-2
Table or domain maximums.....	D-3
Entries per table and OMNIDEX ID value.....	D-3
Number of keyword keys	D-3
Number of keyword key groups.....	D-3
Number of sorted keys	D-3
Key maximums.....	D-4
Multiple keys	D-4
Sorted keys.....	D-4
Glossary	Glossary-1

Using this Guide

Overview

This preface explains the document conventions used in the *OMNIDEX Administrator's Guide* and gives you an overview of each chapter in the book.

Conventions

This guide uses the following conventions:

New terms

Whenever a new term is introduced, it *looks like this*.

USER INPUT

Commands that you enter LOOK LIKE THIS. For example:

```
RUN OMNIUTIL.PUB.DISC
```

After you type the command, always press **[return]** to process that command.

Screen text

Prompts and messages look like this. Prompts are requests for information by a program. Messages return information. For example:

```
Enter the database name:
```

```
The database name is invalid.
```

variables

Information that you must supply, including parameters, *looks like this*. For example:

```
[XL.group.account]
```

Here, *group* and *account* could be any group and account.

[keys] When you are told to press a key on the keyboard, the key name is surrounded by square brackets and **[looks like this]**. For example:

Press **[return]**

Press **[ctrl]-Y**(means press the **[ctrl]** and **Y** keys simultaneously)

How this guide is organized

This *OMNIDEX Administrator's Guide* is organized as follows:

- ❑ This preface tells you the document conventions used in the guide and how this document is organized.
- ❑ "Chapter 1: Introduction", explains the benefits and features of OMNIDEX.
- ❑ "Chapter 2: OMNIDEX Installation", tells you how to install OMNIDEX keys, load the indexes and test your installation.
- ❑ "Chapter 3: Utilities", tells you how to use OmniUtil, DISC's OMNIDEX installation and maintenance utility, DataView, DISC's database query utility, and DBMGR, DISC's database data support tool.
- ❑ "Chapter 4: Topics", discusses a variety of topics that pertain to designing, installing and maintaining OMNIDEX keys.
- ❑ "Appendix A: Data Type Considerations", discusses how OMNIDEX supports and indexes different IMAGE data types.
- ❑ "Appendix B: OMNIDEX Parsing Rules", tells how OMNIDEX parses fields during indexing, how it parses arguments during searches, and how it translates 8-bit extended ASCII characters.
- ❑ "Appendix C: Installing OMNIDEX Procedure Libraries", tells you how to copy and reference the OMNIDEX XLs and SLs to gain indexed access to TurboIMAGE databases.
- ❑ "Appendix D: OMNIDEX Version 3 Limits", discusses the limits and maximums for OMNIDEX, including the number of keys you can install, the length of keys, and so on.
- ❑ "Appendix E: OMNIDEX Searches", describes the kinds of searches that OMNIDEX makes possible.

This manual also includes a glossary and an index.

Technical support

If you have questions about OMNIDEX that are not addressed in this manual, you can get technical support by calling one of these numbers:

	USA and the Americas	UK and Europe
Voice:	(303) 444-6610 (production) (303) 444-4000 (trial)	+(44) 1372-362777
Fax:	(303) 444-0208	+(44) 1372-386418
Email:	support@disc.com	support@disceurope.co.uk
Address:	5733 Central Ave. Boulder, CO 80301	25-29 High St. Leatherhead Surrey, England KT228AB
Hours	7:30 AM - 6:00 PM MST	9:00 AM - 5:30 PM UK time

Acknowledgments

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.
- ❑ ASK and MANMAN are registered trademarks of Ask Computer Systems Inc.
- ❑ The following are registered trademarks of Cognos Corporation: Powerhouse, QDD, QTP, QUICK and QUIZ.
- ❑ Excel is a registered trademark of the Microsoft Corporation.
- ❑ 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, IMAGE/SQL, MPE V, MPE XL, Native Mode, PASCAL/3000, Query, Rapid, RPG, SORT, SPL, TRANSACT/3000, and TurboIMAGE.
- ❑ Lotus 1-2-3 is copyrighted by, and Symphony is a registered trademark of, Lotus Development Corp.
- ❑ MCBA is a trademark of the Orion Group.
- ❑ MPEX is a trademark of VESOFT.
- ❑ MULTIVIEW GL is a registered trademark of the Multiview Corporation.
- ❑ NETXFER is a registered trademark of Telamon.
- ❑ The following are registered trademarks of Robelle Consulting Ltd.: SUPRTOOL and Qedit.
- ❑ Speedware is a registered trademark of Speedware Corp.
- ❑ The IMAGE/3000 Handbook is copyrighted by Wordware.

Chapter 1: Introduction

This chapter provides a general overview of OMNIDEX, its many powerful search capabilities, and some related software products. It is divided into the following sections.

- ❑ “Welcome to Version 3 of OMNIDEX”, on page 1-2, provides a general introduction to OMNIDEX's access capabilities.
- ❑ “Why Use OMNIDEX?”, on page 1-8, describes a few of the benefits that OMNIDEX provides to you and your users.
- ❑ “Implementing OMNIDEX”, on page 1-12, describes the four easy steps to installing keyword and sorted key access on your databases.
- ❑ “Components of OMNIDEX”, on page 1-13, describes the programs, utilities and procedure libraries that OMNIDEX comprises.

Welcome to Version 3 of OMNIDEX

Welcome to version 3 of the OMNIDEX indexing system, the powerful enhancement to TurboIMAGE. OMNIDEX uses indexes for extremely fast and flexible information retrievals. You can index (or key) virtually any number and type of fields in a database, including fields in master data sets. You also can design composite keys from parts of fields. OMNIDEX indexes are updated automatically whenever you add, delete or change the data in a keyed field.

OMNIDEX provides two types of keys to speed access to your data: keyword keys and sorted keys. Each of these two types of keys provides a different set of access capabilities. Keyword keys are discussed next. Sorted keys are discussed later.

What are keyword keys?

Keyword keys support keyword searches to find records. When you key an item for keyword access, individual data values in that field (delimited by spaces, commas and other special characters) are parsed and indexed as *keywords*. When you use words as arguments in a search on a keyword key, the TurboIMAGE TPI¹ or ImagePlus intrinsics search the indexes to instantly locate all records that contain those words. This enables you to instantly locate information using words or values contained *anywhere* in a keyword field.

Keyword keys give you the flexibility to:

- retrieve master records without knowing their SI values
- retrieve detail records directly, without chained reads
- retrieve records by searching several keyed fields simultaneously
- search across sets to qualify a master and its related detail records
- search across sets in different databases

1. TPI refers to the Standard Interface to Third Party Indexing.

Keyword keys also support the following search operations:

- range retrievals
- relational retrievals (like greater than and less than)
- Boolean searches (AND, NOT, OR)
- generic (partial keyword) and pattern-matched (wildcard) searches

Searching with keyword keys

Three CUSTOMERS master records are shown in Figure 1-1 on the next page. The underlined fields (COMPANY, CONTACT, STATE and COMMENTS) are keyword keys and the CUSTOMER-NO field is the TurboIMAGE search item (SI) of CUSTOMERS.

To quickly retrieve any master record with TurboIMAGE access alone, you must know its SI (CUSTOMER-NO) value. You can retrieve master records based on the contents of a data field (like COMPANY), but that involves a serial read, which can take a while.

<p>CUSTOMER-NO: 1 <u>COMPANY:</u> Dynamic Information Systems <u>CONTACT:</u> Dave Smith <u>ADDRESS:</u> 5733 Central Ave <u>CITY:</u> Boulder <u>STATE:</u> CO <u>ZIP-CODE:</u> 80301 <u>COMMENTS:</u> aka DISC, software development company. DBMGR Omnidex, OmniView, OmniQuest OmniWindow. Sold Internationally.</p>	<p>CUSTOMER-NO: 2 <u>COMPANY:</u> Information Xpress <u>CONTACT:</u> Dave Jones <u>ADDRESS:</u> 2001 Hitek Blvd. <u>CITY:</u> Broomfield <u>STATE:</u> CO <u>ZIP-CODE:</u> 80020 <u>COMMENTS:</u> software consulting firm. Designs Omnidex application programs for manufacturing inventory and distribution.</p>	<p>CUSTOMER-NO: 3 <u>COMPANY:</u> Dynamic Data Storage <u>CONTACT:</u> Joan Smith <u>ADDRESS:</u> 931 Hennepin Ave. <u>CITY:</u> Minneapolis <u>STATE:</u> MN <u>ZIP-CODE:</u> 55455 <u>COMMENTS:</u> aka DDS. Hardware manufacturer of disk drives.</p>
---	---	---

Figure 1-1: Three OMNIDEX master records

With OMNIDEX installed, you can perform keyword searches on keyword keys to find any master record instantly. For example, if you searched a keyword key installed on COMPANY using the keyword argument INFORMATION, you would find all CUSTOMERS records with the word "information" somewhere in the COMPANY field. This is called a *keyword retrieval*.

In an online application, the search might look like this:

Enter a company name: INFORMATION

Records qualified: 2

List? Y

Customer number	Company
1	Dynamic Information Systems
2	Information Xpress

Records 1 and 2 qualify because they both have the word "information" somewhere in their COMPANY field. It doesn't matter where a keyword occurs in a key field. Keyword searches are not position sensitive.

You can install keyword keys on free form text fields, like the COMMENTS field in the sample records in Figure 1-1. The next search uses a combination of keywords to search against the keyword key installed on COMMENTS. The Boolean AND operator tells OMNIDEX to qualify records with the words "software" and "development" in the key field being searched (COMMENTS).

```
Comments: SOFTWARE AND development
```

```
Records qualified: 1
```

```
List? Y
```

```
Customer number    Company
```

```
1                  Dynamic Information Systems
```

The search qualified the only record with both "software" and "development" in the COMMENTS key field. Notice that the arguments were entered in upper and lower case. Keyword searches are not case sensitive by default, but can be if you want.

You can also perform keyword searches across fields. For example, OMNIDEX lets you search for records that contain "dynamic" in their COMPANY field, and "MN" in their STATE field. In the prompting program, the search might look like this:

```
Enter a company name: INFORMATION
```

```
Records qualified: 2
```

```
List?
```

```
Enter the state: MN
```

```
Records qualified: 1
```

```
List? Y
```

```
Customer number    Company
```

```
3                  Dynamic Data Storage
```

Of the two records with "dynamic" in the COMPANY field, only one record (record number 3) has "MN" in its state field. Notice how OMNIDEX returns a qualifying count after each keyword search. Sometimes, this count is all the information you need.

Keyword keys are discussed in greater detail in the "OMNIDEX Retrievals" section of the "Topics" chapter.



What are sorted keys?

Sorted keys enable you to retrieve records in order, sorted numerically and alphabetically by key value. The records can be retrieved in either ascending or descending sorted order. This is useful for a variety of applications, from generating mailing labels sorted by zip-code, to listing transactions sorted by department and date.

These sorted access capabilities can be installed immediately on an existing database, without modifying the database or creating external files to store records redundantly. When you key a field for sorted access, an index is created to store sorted key values and record identifiers.

The use of indexes reduces the data redundancy usually associated with maintaining external files of sorted records, and the system overhead of maintaining pointers for sort paths. It also means that OMNIDEX sorted keys are faster, and more efficient to use than other means of sorting records. These indexes are protected by all of TurboIMAGE's security features.

Sorted keys support the following search operations:

- range retrievals
- relational retrievals (like greater than and less than)
- partial-key searches

Searching with sorted keys

Unlike keyword searches, sorted searches *are* position sensitive. You must supply the first few (most significant) bytes of a key value to search for records using a sorted key.

If you installed a sorted key on a ZIP-CODE field, the values for that field would be stored in sorted order in the index created for that key.

When you supply an argument in a search on a sorted key, the intrinsics compare your argument, byte for byte, against the key values stored in the index. The intrinsics then return records sorted in key order (by ZIP-CODE) to your application program.

Sorted keys also support partial-key (generic) searches, which are especially useful when searching on hierarchical code fields. Account numbers are often hierarchical fields where the first few bytes represent a regional code, the next few bytes a SIC code, and so on. When a sorted key is installed on a hierarchical code field, it lets you find records as generally or as specifically as you want, depending on how much of the key value you specify in your search argument.

For example, if you supply a partial argument, like 80, you would find records where the first two bytes of the ZIP-CODES match the two-byte argument.

A report of the qualifying records might look like this:

Customer Number	Company	Zip Code
2	Information Xpress	80020
1	Dynamic Information Systems	80301

Sorted keys are discussed in greater detail in the “OMNIDEX Retrievals” section of the “Topics” chapter.



Why Use OMNIDEX?

Three thousand sites world-wide use OMNIDEX. Some of the reasons are:

- speed of access to data
- retrieval flexibility
- ease of installation and maintenance
- faster application development
- reliability

Each of these reasons are discussed next.

Speed of access to data

Probably the biggest reason to use OMNIDEX is the speed with which it can locate records. With TurboIMAGE you don't have to worry about speed *as long as you always search on key fields using full key values*. If you search on a field that is not a TurboIMAGE key you have to wait while your application program serially reads every record and vacant address in the data file. If you have half a million master records in your data file, the search would take 55 minutes², whether it qualified five records or five thousand records.

If you were looking for a specific topic in a book, you wouldn't read the book from cover to cover until you found the topic. Instead, you would look for the topic in the index and go directly to the pages that contained information about that topic. Because OMNIDEX uses indexes to locate data, it eliminates the need for serial reads.

-
2. All examples assume the blocking factor is six and the disk access time is thirty IOs per second.

OMNIDEX version 3 qualifies between 40,000 and 225,000 records per second, depending on CPU class. To qualify five hundred records, then, would take less than a second! To return the qualified records would require another 500 IOs, one for each DBGET, bringing the total retrieval time to 17 seconds!

This savings in time and system overhead means that you can get your data right away, in real time. It also means increased throughput, and a more efficient use of system resources.

1

Retrieval flexibility

OMNIDEX enables you to key any number of fields in any set, including TurboIMAGE masters for instant keyword retrieval and sorted access. You can install these keys on an existing database in minutes, without restructuring it. OMNIDEX lets you access any item in your database as though it was a key item.

OMNIDEX lets you index and retrieve by individual words or values (keywords) in a field. This means you can find a CUSTOMERS record, for example, without knowing a TurboIMAGE SI value. 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. This capability is invaluable to customer service representatives, as well as other users.

Sorted keys let you retrieve records in order, sorted by key values. If you create a composite key that combines several fields, you can sort records by several fields. This enables you to generate reports, like mailing labels, or general ledgers, that are sorted for easy reference.

As you learn more about OMNIDEX's many features and discover new uses for them, you can easily add them to your existing installation. Your users will like OMNIDEX's many access features, which let them ask for data in their own language instead of TurboIMAGE key values. OMNIDEX also supports native language extended character sets (like Roman8 and Turkish8).

Ease of installation and maintenance

OMNIDEX's OmniUtil installation utility lets you install OMNIDEX keys on your database, or change how they are installed, in a matter of minutes. OmniUtil is very easy to use, and contains online help at each prompt. When you are finished specifying keys for installation, OmniUtil creates the required indexes automatically at your convenience.

The data integrity of the OMNIDEX indexes is maintained automatically. If your database is enabled for Third Party Indexing under MPE/iX version 4.0, all TurboIMAGE updates automatically maintain the OMNIDEX indexes as well. If your database is not enabled for Third Party Indexing, the OMNIDEX intrinsics update the indexes to reflect any data that is added, updated or deleted in your database. Your current data entry programs can be run, unchanged, through a *Call Conversion library* to automatically update the indexes whether they are enabled for TPI or not.

The most work to installing OMNIDEX is in deciding what keys to install on your database. "Determining Retrieval Requirements", on page 4-28 of the "Topics" chapter, contains all the information you need to decide how to install OMNIDEX.

Faster application development

OMNIDEX version 3 gives you the choice to use the Standard Interface to Third Party Indexing products, or the traditional OMNIDEX Intrinsic Interface.

The Standard Interface to Third Party Indexing (TPI)

If you use TurboIMAGE version C.04.03 or later, you can perform OMNIDEX retrievals and updates by calling TurboIMAGE/iX intrinsics at the system level. Because this interface to OMNIDEX is integrated at the system level, it makes the best use of system resources and optimally manages concurrent operations.



Please run the TPIXLCFG.UTIL.DISC command file if you are accessing TPI-enabled databases.

The OMNIDEX Intrinsic Interface

OMNIDEX intrinsics are similar to TurboIMAGE intrinsics, so they are easy to learn and incorporate into your programs. They can be used by many software packages and programming languages. For example, you can create QUICK screens that use OMNIDEX's powerful access capabilities.

Because the OMNIDEX and TPI 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 programs. Similarly, OMNIDEX and the Standard Interface to Third Party Indexing eliminate the need for much of the data manipulation, like upshifting and truncating, from retrieval programs.

Dynamic Information Systems Corporation (DISC) can provide specific information, and in many cases, procedure source code examples, for your programming language. To add OMNIDEX retrieval capability to existing online programs without programming changes, you might consider OmniWindow. OmniQuest, available from DISC, provides instant OMNIDEX access from report writers and transaction processors like QUIZ, QTP and QUERY, or from your batch programs. Complete application interfaces to OMNIDEX are also available for ASK/MANMAN, Multiview, MCBA, Growth Power, and Gerber-Alley.

For more information about developing OMNIDEX applications, see the *OMNIDEX ImagePlus SDK API Guide*.

Reliability

OMNIDEX has been in use since 1985 and is installed on over three thousand sites world wide. Many third party vendors have included OMNIDEX into their information management software products. DISC's product development staff continually refines OMNIDEX to make it easier to use and better suited to your needs.

Implementing OMNIDEX

There are four easy steps to implementing keyword and sorted access capabilities on your TurboIMAGE database.

1. Decide how to install keyword and sorted keys.

This is the most important step in the process of implementing OMNIDEX. The “Topics” chapter discusses the things you should consider to make the best use of OMNIDEX.

2. Install keyword and sorted keys.

When you have decided what keys you want to install on your databases, install them using OmniUtil. OmniUtil creates the indexes necessary for keyword and sorted access. See chapter 2, “OMNIDEX Installation”, for detailed information about installing OMNIDEX.

You can also use OmniUtil to change the way OMNIDEX is installed on your database, and then reindex only the keys you changed.

3. Load the indexes with key values.

Use OmniUtil to load the indexes either online or in batch. Because this step may require exclusive access to a database, you use OmniUtil to create an indexing job. Then you can stream or schedule the job to load the indexes at your convenience. See chapter 2, “OMNIDEX Installation”, for more information about loading the indexes.

4. Develop new applications to query and update the OMNIDEX-enhanced database.

To use OMNIDEX's powerful access capabilities, you can modify existing programs or develop new ones. While you develop retrieval applications, you can use DataView to test your OMNIDEX installation and find records. See the *OMNIDEX ImagePlus SDK API Guide* for information about developing retrieval applications.

Components of OMNIDEX

The OMNIDEX indexing system has the following components:

- ❑ OMNIDEX utilities
- ❑ OMNIDEX intrinsic libraries

These components, which are listed below according to function, support the complete implementation and maintenance of OMNIDEX on your databases.



The OMNIDEX utilities

OmniUtil

OmniUtil is a program that combines all the OMNIDEX installation, indexing, configuration and maintenance operations into one menu driven interface. The main menu screen appears in Figure 1-2 on the next page.

You can use OmniUtil to:

- ❑ install and maintain OMNIDEX keys on a database
- ❑ generate and schedule jobs to perform routine maintenance tasks
- ❑ display information about a database or the keys installed on it, as well as OMNIDEX version information
- ❑ get help with errors incurred while using DISC products

Because it is menu-driven, OmniUtil is very easy to use. Function keys and cursor activated “buttons” help you navigate through menus and operations. OmniUtil also contains extensive online help to guide you through installing and maintaining OMNIDEX keys. See the “OMNIDEX Installation” and “Utilities” chapters for detailed information about OmniUtil’s menus and its use in installation and index maintenance.

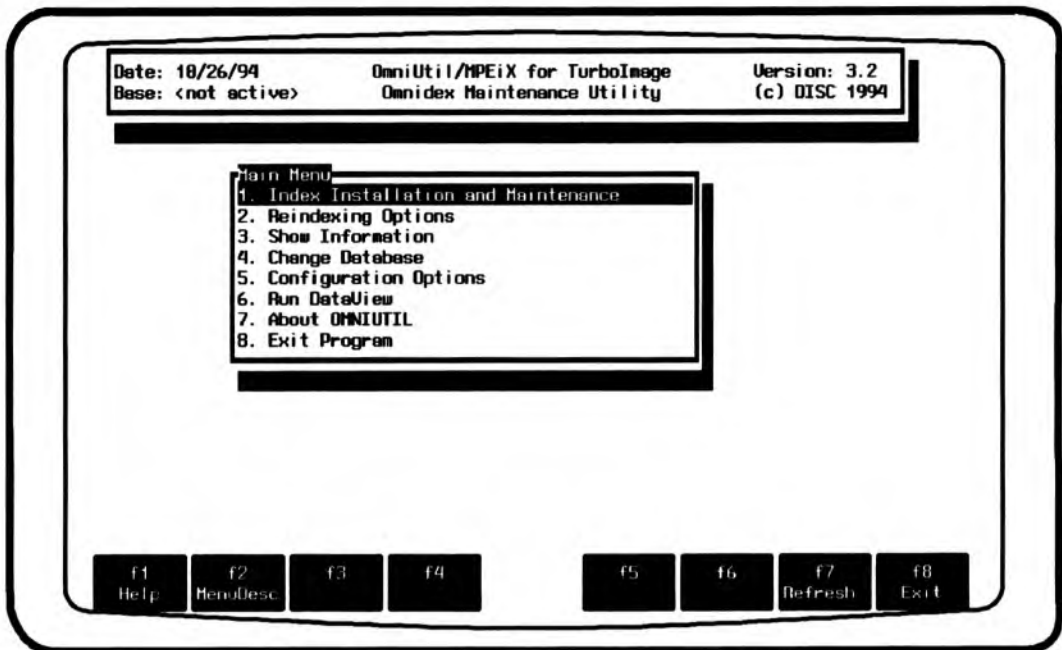


Figure 1-2: The OmniUtil Main Menu

DataView

DataView is a database query tool designed for OMNIDEX databases. DataView lets you retrieve entries by keyword or sorted keys. It is ideal for testing the OMNIDEX keys you've installed on a database, and can provide OMNIDEX access while you develop more specialized data retrieval applications.

You also can run other programs, as well as issue UDCs and MPE commands, from within DataView. Task Manager, which was developed for OmniWindow, lets you suspend DataView to run other applications or execute commands. After you are finished, Task Manager lets you save that command or application to a task list.

Anyone can learn to use DataView. It uses an intuitive approach to finding, listing, and viewing records, and has extensive online help. Function keys and cursor activated "buttons" make navigating through DataView easy. See "DataView", on page 3-21, for detailed information about DataView's menus and functions.

DBMGR

DBMGR is DISC's database support tool. DBMGR gives you the capability to perform several database maintenance operations under one utility. These operations include data set capacity management, copying and renaming data sets, erasing and reloading data sets, and deleting data sets and data items.

Intrinsic libraries

Intrinsics are callable procedures used in your application programs for data retrieval, updating records and other functions related to your TurboIMAGE database. Several intrinsic libraries are included in OMNIDEX:

- ❑ the OMNIDEX intrinsic library
- ❑ the Call Conversion library
- ❑ a switch stub intrinsic library

Each of these libraries are discussed briefly below, and in more detail in the "Programming" chapter of the *OMNIDEX ImagePlus SDK API Guide*.

The OMNIDEX intrinsic library

The OMNIDEX intrinsics, which reside in XLOMNIDX.PUB.SYS, call the TurboIMAGE intrinsics in XL.PUB.SYS. If a database is enabled for Third Party Indexing (TPI), TurboIMAGE/iX intrinsics call the OMNIDEX ImagePlus intrinsics to perform functions related to the indexes.

The ImagePlus intrinsics differ from TurboIMAGE intrinsics in several respects.

- ❑ The parameters they use differ slightly from TurboIMAGE intrinsic parameters.
- ❑ They perform a few more functions, like transferring record identifiers.

The OMNIDEX sorted access intrinsics, and keyword access intrinsics, and general intrinsics reside in the same library, but are separate components of OMNIDEX. Keyword and sorted keys use different indexing methods to provide their different retrieval capabilities. Therefore, DISC developed two different sets of access intrinsics, one for keyword access, the other for sorted access. A third set of intrinsics, general intrinsics, performs database maintenance functions. Each set of intrinsics is discussed below.

Keyword access intrinsics are called by your application programs to provide keyword retrieval capabilities, which were described earlier. They also enable you to:

- ❑ transfer retrieved information to a file
- ❑ obtain information about any keyword keys installed on a database

The *sorted access intrinsics*, DBIFIND and DBIGET, perform all of the sorted sequential retrieval operations, plus any standard TurboIMAGE retrievals.

OMNIDEX *general intrinsics* perform maintenance functions. The general intrinsics:

- ❑ add, delete, and modify records and indexed key values
- ❑ handle OMNIDEX errors
- ❑ return information about an OMNIDEX database's structure and key placement

You can substitute OMNIDEX intrinsics for almost all the TurboIMAGE intrinsics. The exceptions are DBBEGIN, DBEND and DBMEMO, which have no OMNIDEX counterparts.

The Call Conversion library

The Call Conversion library, which resides in XL.PUB.DISC, makes your existing update programs automatically update OMNIDEX indexes. Call Conversion traps calls to TurboIMAGE intrinsics, and converts them into calls to OMNIDEX general intrinsics. For example, a DBPUT mode 1 is converted automatically to a DBIPUT mode 1. DBIPUT mode 1 adds a record entry — just like DBPUT mode 1 — then, automatically updates the OMNIDEX indexes.

Call Conversion is important for TurboIMAGE programs that add, delete, or update records in databases not enabled for TPI. Without Call Conversion, changes to such a database are not reflected in the OMNIDEX indexes, and OMNIDEX retrievals could return erroneous data. If you want to update the OMNIDEX indexes of databases in real time without enabling them for TPI, run your update applications through the Call Conversion library.

The Call Conversion library also traps calls to OMNIDEX intrinsics and directs them to the OMNIDEX Intrinsic library.

The Standard Interface to Third Party Indexing (TPI)

If you are running TurboIMAGE version C.04.03 or later, you can perform OMNIDEX keyword and sorted retrievals using the TurboIMAGE intrinsics that reside in XL.PUB.SYS. The Standard Interface to Third Party Indexing lets you perform retrievals on sorted and keyword keys using familiar DBFIND and DBGET coding techniques. It also updates OMNIDEX indexes automatically. It provides a slightly different set of capabilities, however, and is not portable to other operating systems.

See “OMNIDEX Retrievals”, on page 4-2 of the “Topics” chapter, for more information about the Standard Interface to Third Party Indexing. Appendix D tells how to set up the Standard Interface to Third Party Indexing. The *OMNIDEX ImagePlus SDK API Guide* provides detailed information about both the OMNIDEX Intrinsic Interface and the Standard Interface to Third Party Indexing.

Switch stub intrinsic library

The switch stub intrinsic library lets compatibility mode programs use native mode OMNIDEX software. When switch stubs routines are copied from SL.PUB.DISC or USL.PUB.DISC to your Compatibility Mode application accounts, they intercept intrinsic calls and convert them to native mode calls to the appropriate XL. Compatibility mode applications are discussed in the “Programming” chapter of the *OMNIDEX ImagePlus SDK API Guide*.



Why wait any longer for your data?

In the pages that follow, you will learn how to install all of the features of OMNIDEX on your databases. The OMNIDEX documentation provides all the information you need to implement OMNIDEX.

For those of you who might benefit from hands-on training, DISC offers four day, intensive OMNIDEX training classes. The classes are taught by members of our technical support staff. Training involves instruction combined with extensive lab work in implementing OMNIDEX's capabilities on the student's own applications. Classes are held at our home office in Boulder, Colorado, and at other select locations throughout the United States. For schedules or more information about OMNIDEX training classes, contact your DISC sales representative at (303) 444-4000.

Chapter 2: OMNIDEX Installation

2

This chapter describes the simple process of installing keyword and sorted keys, and their options, on a database. The process is divided into these basic steps:

1. “Installing OMNIDEX Keys”, on page 2-3, involves selecting fields to be keyword and sorted keys, or creating composite keyword or sorted keys from parts of fields.
2. “Activating the Installation”, on page 2-19, involves creating the index structures required to support the keys you installed in step 1, above.

The installation process can include another two steps before loading the indexes. Excluding noise words from keyword indexes, discussed on page 2-22, improves indexing performance. “Customizing the Translation of 8-bit Characters”, on page 2-24, tells you how to change how OMNIDEX indexes 8-bit ASCII characters.

3. “Loading the Indexes”, on page 2-28, loads key data into the indexes that were created in step 2, above. This step can take some time to complete. So you may want to load the indexes in batch. See “Indexing jobs”, on page 2-30, for more information.
4. “Verifying and Testing the Installation”, on page 2-32, involves using the newly installed OMNIDEX keys to find and retrieve records.

Before you begin

Before you use OmniUtil to install keyword or sorted keys on a database, the following conditions must exist.

- The database that you plan to key for OMNIDEX must exist.
- The database cannot be file equated.
- You must be logged on as the database creator.
- PM (Privileged Mode) capability must be present in the group where the internal utilities reside (usually this is PUB.DISC).



If you plan to load a high volume of records into your database, load the records before you install OMNIDEX keys.

Although you can change how keys are installed on a database, you should decide where to install keys before you begin an installation.

To begin installing OMNIDEX keys on a database, run OmniUtil. To run OmniUtil enter the following statement at a system prompt:

```
RUN OMNIUTIL.PUB.DISC
```

Remember to press **[f1]** for help, and **[f2]** for a description of any menu item. The next section describes how to install keys on your database.

Installing OMNIDEX Keys

From the OmniUtil Main Menu (shown in Figure 3-1, on page 3-6), select 1. Index Installation and Maintenance. From the Index Installation submenu, select 1. Install or Modify Indexes.

OmniUtil prompts you for a database. Either enter the name of a database, or press **[f2]** and select a database from the pick list that OmniUtil displays.

2

Selecting a table (data set)

When OmniUtil prompts you for a table (data set), either enter the name of a table, or press **[f2]** and select a table from the pick list that OmniUtil displays.

When you select or enter a table, OmniUtil displays its name and highlights a prompt that reads `Add/Modify Indexes`. Press **[return]** to begin adding keys. If you have already installed keys on the table, they are displayed in a window labeled `Accepted Fields`. To get information about these keys, select `View Indexes` from the Table window.

After you select a table, OmniUtil opens a field prompt window, where you can key existing fields or create composite keys. Keying fields and creating composite keys are discussed in the next two sections.

When you are finished installing keys on a table, you can return to the table window either by pressing **[f8]** (labeled `Done`), or by pressing **[return]** when prompted for another field.

Keying fields

When you select `Add/Modify Indexes` for a given table, you can install keys on that table. OmniUtil prompts you for a field to key with OMNIDEX access. Either enter the name of a field, or press **[F2]** and select a field from the pick list that OmniUtil displays.

When you select or enter a field, OmniUtil displays its name in the Field window, and moves the highlight bar to let you select a key type. You can select either `Multiple`, `Sorted`, or `Multiple & Sorted` to designate the type of access (or key) to be installed on the selected field.

Creating composite keys

Composite keys are keys that you create from parts of fields. You can create composite keys that combine fields or parts of fields, isolate part of a field, or reorder a field. Composite keys are discussed in detail in “Designing composite keys” on page 4-43 of the “Topics” chapter.

When OmniUtil prompts for a field and you enter a name that does not correspond to a field in the database, OmniUtil asks if you want to create a composite key with that name. If you select `(Yes)`, OmniUtil begins the process of creating a composite key by prompting you for components.

When asked to supply a component, you can supply the name of any field in that table by entering it in the Component prompt window, or by pressing **[F2]** and selecting its name from the pick list.

Because composite keys can comprise parts of fields, OmniUtil prompts for a start byte and length after you supply a component field's name. To use the entire field in the composite key, simply press **[return]** at the `Start Byte` prompt, and OmniUtil prompts for the next component. To use only part of a field in the composite key, you must supply the start byte and length for that component. For example:

- ❑ To use bytes 5 through 8 of a field as a component in the composite key, enter 5 at the `Start Byte` prompt, and 4 at the `Length` prompt.
- ❑ To use the first four bytes of a field as a component in the composite key, enter 1 at the `Start Byte` prompt, and 4 at the `Length` prompt.
- ❑ To use an entire field as a component in the composite key, press **[return]** at the `Start Byte` prompt.



You can use entire binary fields as components in composite keys, but not parts of binary fields.

When you specify an integer (type I, J, or K), packed (type P), or real (type R) field as a component in a composite key, OmniUtil accepts the entire field as a component and does not ask for a start byte and length.

To stop adding components to a composite key, press **[return]** in the Component input field. OmniUtil displays the composite key's name in the field window, and moves the highlight bar to let you select a key type. You can select either **Multiple** or **Sorted**, but not both. If you select both, OmniUtil reminds you:

Composite keys must be either Multiple or Sorted, but not Both.
Ok. (Help)

Figure 2-1: Composite key type warning

Typecasting composite key components

When specifying composite key components, you can also specify the “type” of the component. This determines how that component will be stored. Specify a type for a component whenever the type of data in a TurboIMAGE field is not accurately described by the item definition in the TurboIMAGE schema.

To specify the components type, enter the type in the Component prompt window, shown below:

No.	Component	Column Name	Start Byte	Length	Type
1.	CUSTOMER-BAL		1	4	X

Figure 2-2: The Type option in the Component prompt window

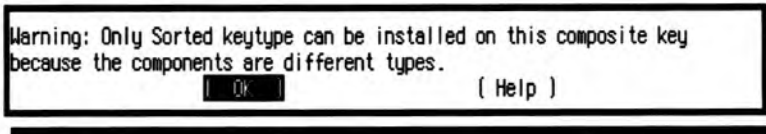
The valid types that you can assign to a key are:

- I or J, to index the component as signed binary values
- K, to index the component as unsigned binary values
- P, to index the component as packed values
- R, to index the component as real values
- E, to index the component as IEEE real values
- X, to index the component as character values
- Z, to index the key as Zoned values



When you typecast a character component as binary (I, J or K), the number of bytes in the component must be even.

If the components are of different types, the following dialog box appears.



If OmniUtil does not return any warnings, the key will support either keyword or sorted access. The effect of data type and length on OMNIDEX access is discussed next.

Defining composite keyword keys

When creating composite keys with a single component for keyword key retrievals, the component can be of ASCII or binary type. A binary component must be specified as the entire binary item. That is, you cannot split binary fields. To install keyword access on a composite key that contains several components, all components must be of the same data type.

The example below shows a composite keyword key that composed of a 4 byte integer extracted from a 6 byte character (X6) field.

Component	Data type	Start byte	Byte length	Type=	Resulting data type
CUST-BAL	X6	3	4	I	I2

Defining composite sorted keys

When creating composite keys for sorted retrievals, components can be a mix of ASCII and binary types. When a composite key is specified with a mix of data types, OmniUtil warns that the field can only be a sorted key, as shown on page 2-6.

Adding key options

Key options let you tailor retrieval and update capabilities for any key, including composite keys. Some options only apply to keyword keys. Other options apply to both keyword and sorted keys.

Adding key options is the last step when keying a field or creating a composite key. After you assign a type (Multiple, Sorted, or Multiple & Sorted) to a key or composite key, a menu appears. At this menu you can do one of three things.

- Choose (OK) to install the key without options.
- Choose (Options) to assign options before installing the key.
- Choose (Cancel) to reject the key and start over.

When you choose (Options), a window similar to the one in Figure 2-3 appears with a set of key options based on the key type you chose.

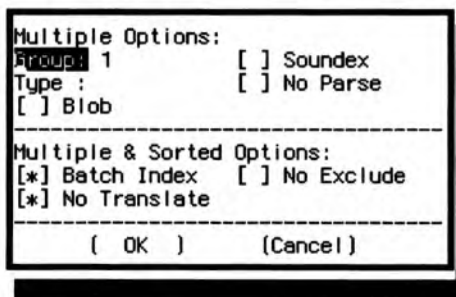


Figure 2-3: The options window for a keyword & sorted key

To select an option for a key, highlight it using your cursor keys and press **[return]**. An asterisk (*) displayed next to an option (or a number displayed next to Group:, or a letter displayed next to Type:) means that it's been selected.

To remove a selected option, highlight it and press **[return]** again.

When you have selected all the options for a key, you can either move the highlight bar to (OK) and press [return], or press [f5] to accept the options. To cancel installing options, you can either move the highlight bar to (Cancel) and press [return], or press [f6].

The options, how to assign them, and what they do are discussed next.

Keyword key options

The keyword key options are:

- Blob indexing
- Grouping
- No Parse
- Type Casting
- Record Complex (detail tables only)
- Soundex

Note that you cannot install these options on individual components of composite keys.

Blob Indexing

By default, when you install a keyword key on a TurboIMAGE compound item, OMNIDEX indexes each element of the item as an individual key and groups the elements together. Keywords in an array, then, are parsed not only by spaces and special characters, but also by element boundaries. This prevents words that span element boundaries, or “wrap around”, from being properly indexed.

The Blob indexing option lets you treat TurboIMAGE compound items as one OMNIDEX key field, regardless of element boundaries. This option is useful when words span element boundaries, as in the example below.

<pre> TURBOIMAGE-NO DATE-ENTERED ENTERED-BY DESC-LINES (2) (3) (4) (5) </pre>	<pre> 324 R20027 HEN Have development projects coming up. Will be ready to evaluate the software in about a month. In the meantime, end of quarter concerns are too pressing. </pre>
---	--

Note, however that if one word ends on a field boundary and another begins on the next field boundary, Blob indexing combines the two words into one keyword.

To install Blob indexing on keyword keys, select the [] Blob item in the options menu.

Grouping

Grouping is a very powerful feature that lets you combine several keyword keys into one logical retrieval unit, as if they were one key. Grouping can reduce the number of intrinsic calls required when searching for records. For example, if the keyword keys COMPANY-NAME, CITY, and STATE are grouped together, you can search all three keys with one intrinsic call.

In online applications, you can search several grouped keys at a single prompt. For example, to find records for Hewlett-Packard in Roseville, California, you could enter `Hewlett, Roseville, CA` as a keyword argument. The search might look like this:

```
Company Info? Hewlett, Roseville, CA
1 entries qualify
```

If COMPANY-NAME, CITY and STATE were not grouped, you would need a separate intrinsic call for each key. This would not noticeably affect performance, but would require more intrinsic calls to find records.

In an online application, a search on the ungrouped COMPANY-NAME, CITY and STATE keys might look like this:

```
Company Name? Hewlett
400 entries qualify

Address?

City? Roseville
10 entries qualify

State? Ca
1 entries qualify
```

The result is the same: one record qualified. The grouped keys, however, required fewer intrinsic calls and less programming effort.

To group fields together, select `Group:` from the options menu. Then, in the highlighted area to the right of the `Group:` option, assign the same number for each of the keys you want to include in a single, grouped key. Press **[F5]** or **[return]** after assigning the group number.

To change the group where a key is installed, as when modifying an OMNIDEX installation, assign the key the number of a different group. To remove a key from a group and index it individually, assign it a group number of zero (0).

You can group keys across linked data sets. If a detail table contains keys assigned to a group numbered "1", and you link it to a master table with keys assigned to a group numbered "1", the two groups are combined into one. Thus, to group keys across tables, assign them the same group number, and link their tables. Linking tables is discussed later in this section, and in "Linking Details to Masters" in the "Topics" chapter.

The following principles apply when using the Grouping option.

- Assign group numbers from 1 to 63. You can have no more than 63 groups per OMNIDEX domain (an unlinked master set, unlinked (DR) detail set, or a master set and all detail sets linked to it).
- Do not group ASCII keys with binary keys because ASCII keywords may be interpreted as binary, or vice versa, during a search.
- Group No Translate (NT) keys only with other NT keys.
- Group Record Complex (RC) keys only with other RC keys.
- Do not group keys that contain similar data. For example, if a ZIP-CODE field is grouped with an ADDRESS field, a generic search for all zip codes that begin with the digits "14" would also find records whose addresses begin with the digits "14".

No Parse

The No Parse option disables the parsing of keywords in the OMNIDEX indexes created for keyword keys. Parsing uses spaces, nulls, and other characters (like / - ' ,) in a key to delimit keywords. When the indexes are updated, these spaces and other characters are removed and the values between them are indexed as keywords. "Leonard,Feinstein&Fitch", for example, would be indexed as three separate keywords: "LEONARD" "FEINSTEIN" and "FITCH".

If you install the No Parse option on a key, the first 32 bytes of a keyed field - including special characters, control characters and embedded spaces - are indexed as one keyword. Any characters after the first 32 are not indexed. Parsing is discussed in Appendix B.

If the data in a field contains special characters, you should consider how its data will be indexed. Sometimes, a key may need the No Parse option to keep certain characters from being parsed out of the indexes.

The No Parse option is useful for fields like parts codes that include hyphens and slashes, or for phone numbers that include hyphens and parentheses. When you search No Parse keys, you can enclose in quotes arguments that contain special characters, as discussed on page 4-9.

To install the No Parse option on a keyword key, select [] NoParse from the options menu. Install the No Parse option on character fields only. No Parse is the default for keyword keys installed on integer items (I, J or K), zoned items (Z), search items in master sets, or mixed composite keys.

Type Casting

Type Casting enables you to inform OMNIDEX when data is stored in a format that does not match the TurboIMAGE data type for a keyed field. To typecast a keyword key, select `Type:` from the options menu. When you press **[return]**, a highlighted field appears beside `Type: .` Enter one of the following values:

- I or J, to index the key as signed binary values
- K, to index the key as unsigned binary values
- P, to index the key as packed values
- R, to index the key as real values
- E, to index the key as IEEE real values
- X, to index the key as character values
- Z, to index the key as Zoned values

You can install the Type Casting option on binary fields that store ASCII data, or vice versa. For example, an integer value may be stored in a TurboIMAGE field of type X4. By default, OMNIDEX would assume that ASCII characters were stored in the field. By assigning a type of I when installing the field as a keyword key, key values are then indexed as binary numbers instead of ASCII characters. See "Which fields make good keys?", in the "Topics" chapter, for more information.

To ensure correct searches on ASKDATE, PHDATE, and JDATE fields, typecast them as K if they are not defined as such.

Type Casting also supports the indexing of binary arrays (like a 4J4), provided each element of the array is not longer than four words. By creating a composite keyword key from each element in a binary array, and typecasting it as I, J or K, you can key them for keyword access. You can then group the composite keys.

When indexing a field as a different data type, you must follow the rules that apply to the resulting data type. For example, you could typecast an X8 numeric field as type I, because the field length would not exceed the four word maximum length applied to integer fields. You could not, however, typecast an X12 field as a type J field, because the value stored in that field would exceed the maximum length applied to integer fields.

Similarly, when values are indexed as a different data type, other key options can be affected. For example, if you typecast an ASCII field to be indexed in binary form, you could not group it with another ASCII field. Although the field is ASCII, its key values would be in binary, non-parsed form.

Record Complex

This option is only available for keyword keys installed on detail data sets that are linked to a master set. Installing the Record Complex (RC) option on a keyword key causes OmniUtil to index keywords for an entire detail chain, and not individual detail records, as if it were part of a master. Searching a Record Complex key returns a record complex of one master and its associated details, as discussed in "Linking Details to Masters", on page 4-56.

To install Record Complex indexing on a key, select [] Record Complex from the options menu.

Install the Record Complex option on any keys in a detail that you are grouping with keys in a linked master. This insures that the record identifiers returned for a keyword search are the same for all keys in the group. Do not install this option if you need to select individual detail records, as opposed to entire chains of detail records. Note that keyword keys in masters are installed with the Record Complex option be default.



In OMNIDEX version 2, record complex indexing was the default for keys installed on detail data sets linked to masters. In OMNIDEX version 3, record specific indexing is the default. Note that record complex indexing is preserved on databases that are converted from version 2 to version 3 using OmniUtil.

Soundex

The Soundex option supports phonetic keyword searches. Install Soundex on any field (like name fields) where spelling may cause problems for your users.

For example, when Soundex is installed on a NAME field, the argument ALLEN! finds records with keywords like ALLEN, ALEN, ALAN, and ALAINE. The exclamation point (!) is the Soundex operator. When appended to a keyword argument, it tells the search intrinsic to use the argument in a Soundex (phonetic) search.

You can use several Soundex arguments in a single search. For example, to search for Allen Anderson without knowing the spelling (Alan Andersen? Allen Andersohn?), you could enter the following:

Contact: ALLEN! AND ANDERSON!

To install Soundex on a keyword key select the [] Soundex item from the options menu.

The following principles apply when using the Soundex option.

- You can install Soundex only on parsed character fields. This means you cannot install Soundex on a binary field or a character field where the No Parse option is installed.
- Soundex indexes can require twice the disk space of a normal keyword key. Install Soundex only where needed.
- If you group Soundex keys with non-Soundex keys, be sure to reference the Soundex key in the *field* parameter of ODXFIND when performing a Soundex search.

Sorted key options

There is only one option that applies exclusively to sorted keys, the Unique Key option. The Unique Key option, when installed on a sorted key, imposes a constraint on that key that prevents duplicate values from being indexed for that key. It enforces the uniqueness of each value indexed for the key. If an application tries to post a duplicate value to the sorted index, it incurs error 143 through the ImagePlus API, or error 3143 through the TurboIMAGE interface to Third Party Indexing (TPI).

To install the Unique Key option on a key, select the [] Unique Key item from the options menu.

Keyword and sorted key options

Some key options apply to both keyword and sorted keys. They are:

- Batch Indexing
- No Exclude
- No Translate

These key options are described below.

Batch Indexing

The Batch Indexing option disables real time updating of a key's indexes. Batch Indexed keys are indexed only during a reindexing operation.

To install Batch Indexing on a key, select the [] Batch Index item from the options menu.

Use the Batch Indexing option to eliminate the indexing overhead associated with updating indexes when adding, deleting or updating TurboIMAGE records (as through call conversion). For example, if you enter data during the day, any indexes associated with Batch Indexed keys are not updated at that time. This improves update performance. You can then schedule a reindexing job to reload the indexes at night or on a weekend, when performance is less critical.

Install Batch Indexing only on keys that need not reflect the most current data.

No Exclude

When installed on ASCII keyword keys, the No Exclude option disables the excluded word list, and causes the indexing of every word in the key (except for spaces or null entries). Normally, excluded words are not indexed, but the No Exclude option causes them to be indexed for a particular field. For example, the No Exclude option may be useful on a STATE field. That way, OR and ME are indexed even if the excluded word list includes “or” and “me”.

When installed on binary keyword keys, the No Exclude option causes all values to be indexed, including binary zeros.

When installed on sorted keys, the No Exclude option enables the indexing of spaces and nulls. It does not refer to the excluded word list when installed on sorted keys.

To install the No Exclude option on a key, select the [] No Exclude item from the options menu.

No Translate

For both types of keys, the No Translate option enables case sensitive indexing of key values. It disables upshifting during indexing. For example, “George McMasters” would be indexed as “George” and “McMasters”, not as “GEORGE” and “MCMASTERS”. A No Translate key field also would let you distinguish between the state abbreviation “ME”, and the pronoun “me”. The No Translate option also disables any translation table loaded through OmniUtil’s Database Configurations submenu.

To install the No Translate option on a key, select [] No Translate from the options menu.

Use No Translate if you want to store keyword values for a particular field in case sensitive form, or if you want OMNIDEX to ignore default translation or the translation table loaded to handle 8-bit extended character sets.

If you group keyword keys installed with the No Translate option, group them with other No Translate keys.

Finishing the installation for a table

When you have finished installing keys on a table, you can do either of two things to go on to the next table.

- Press **[return]** at the **Field:** prompt. Then press **[return]** again while (**OK**) is highlighted in the Table window.
- Press **[f8]** at the **Field:** prompt. Then press **[return]** while (**OK**) is highlighted in the Table window.

If you are installing keys on a detail set, OmniUtil asks if you want to link it to a master before returning you to the Table window. If you select (Yes), OmniUtil lets you link the detail to any master set with a path to it. "Linking Details to Masters" is discussed in the "Topics" chapter.

If, during the installation, you want to change how keys are installed on a table, you can easily do so. The steps are outlined next in "Correcting Mistakes."

Correcting mistakes

If you make a mistake during installation, you can always correct it. Installation specifications are not implemented until they are activated, as discussed on page 2-19. This section discusses how to correct mistakes during an installation, before it is activated. For information about changing an installation after you've activated it, see "Changing the Installation", on page 2-34.

If you want to add, remove, or change keys for a table, select that table as you normally would. Then, select **Add/Modify Indexes** to get to the Field installation window.

Adding keys to a table

To add a key, follow the normal installation procedures discussed earlier in this section under "Keying fields", on page 2-4, and "Creating composite keys", on page 2-4.

Removing keys from a table

To remove a key, press **[F4]** (Mod Keys). The highlight bar moves into the Accepted Fields window. Select the key you want to remove, and a window appears that displays the key's name, key type, and options. The window has three buttons that let you accept, modify or delete the key. Select (Delete) to remove the key. The key disappears from the Accepted Fields window when it is deleted.

Changing keys for a table

There are two ways to change a key's type or options.

- Press **[F4]** (Mod Keys). The highlight bar moves into the Accepted Fields window. Select the key you want to change.
- At the Field window, enter the name of the key you want to change, or press **[F2]** to select keyed fields from a pick list. During the initial installation, you must type the name of any composite key, however.

When you do either of these things, a window appears that displays the key's name, key type, and options. The window has three buttons that let you accept, modify or delete the key. Select (Modify) to change the key, as discussed in the next two sections.

Changing a key's type

When you choose to modify a key, OmniUtil opens the Field window, which displays the name of the key, its data type, length, and key type (Sorted, Multiple or Sorted & Multiple). The highlight bar highlights (OK) in the menu below the window.

At this point, you can change the key's type (and options, as discussed next). To change the key's type, press [↑], the cursor-up key. The highlight bar moves to Keytype. Press [return] to gain access to the three different key types.

Use the arrow keys to select a key type, just as you would during normal installation. When you change a key type, any options that you installed on a key are voided. This is because the options supported for each type of key can differ. So you may want to re-specify options for a key after changing its type.

To designate the changed key for installation, press [return]. The highlight bar moves to (OK) in the menu below the Field window. If you press [return] again, the key is designated for installation. If you want to add options to the new key, follow the steps outlined in the next section.

Changing key options

To change a key's options, select the key as discussed above. Then, select the (Options) button at the bottom of the Field window. The options menu appears, and you can select from it as described in "Adding key options", on page 2-7.

Activating the Installation

When you have finished installing keys, press either **[return]** or **[f8]** at the **Table:** prompt.

OmniUtil displays the following message and menu:

```
You have specified the index installation
Please choose one of the following:
```

```
1. Activate the Installation
2. Save the Specified Installation
3. Create/Stream Installation and Indexing Job
4. Modify the Specified Installation
5. View Specified Installation
6. Discard the Installation
```

At this time, OmniUtil has recorded all of your key specifications, but it has not built the OMNIDEX index structures that support keyword and sorted access. These index structures are built in your database's group when you "activate" the installation.

You can activate an installation in three different ways. The means to activating your installation are discussed in the next three sections.

2

Real-time activation

Activating an installation typically takes only a few minutes, depending on the number of keys installed and the number of tables affected. You can activate and index an installation while users are accessing the database as long as the OMNIDEX indexes aren't being accessed.

To activate an installation online, choose 1. `Activate the Installation` from the activation menu (shown above). OmniUtil installs the indexes and displays a message:

```
Please wait, creating index structures.
```

Wait until the revolving cursor stops and OmniUtil displays a dialog box that asks you if you want to create an installation job.



Select (`Yes`) to ensure that you have a copy of the installation if you ever have to reinstall.

OmniUtil then tells you that the `Installation` is complete. To proceed with the installation, select (`OK`) from the dialog box. OmniUtil takes you to an indexing menu like the one discussed on page 2-19.

Saving an installation

If you can't activate an installation immediately after specifying keys, you can save it to a file for later activation. To do so, select 2. `Save the Specified Installation`. This saves your installation in a file created in the database group and account. The default name of the file is the database name followed by "CM". For example, the saved installation for a database named "SALES" would be named "SALESCM".

To use the saved installation, choose 4. `Load Saved Installation from File` from the `Index Installation and Maintenance` menu (discussed on page 3-8).

Installation and indexing jobs

You can also activate an installation by using OmniUtil to create and schedule a job. The generated installation and indexing job not only builds the OMNIDEX index structures, but also loads them with data. This means you can schedule the job for a night or weekend, and return to a fully operational OMNIDEX database!

To use this powerful feature, select 3. Create/Stream Installation and Indexing Job from the activation menu discussed on page 2-19. OmniUtil prompts you for the passwords necessary to stream the job. Next, it asks if you want to stream the job or keep it as a file.

If you choose to keep the file, OmniUtil creates it in your database group and account. The default name is the database name plus the letters "II". For example, for a database named "SALES", the job file would be "SALESII".

If you choose to stream the file, OmniUtil prompts you to schedule a time to stream it. Press [return] to stream the job immediately, or enter the STREAM command parameters to schedule a batch job. Any parameters you enter should follow the STREAM command syntax used by MPE.

OMNIDEX index structures

When you install OMNIDEX, information about the location and types of keys is stored in the OMNIDEX root file. This file is named *dbname0A*, where *dbname* is the name of the OMNIDEX database. Up to 259 additional files are created to index the various keys installed on the database. These, too, begin with the OMNIDEX database's name and are numbered sequentially from *dbname0B* to *dbname9Z*. The OMNIDEX root file has a file code of -410. The rest of the OMNIDEX index structures have a file code of -411.

Excluding Words from Indexing

Excluding words from indexing is not necessary to the functioning of your OMNIDEX-enhanced database. It does, however, result in a savings of disk space, and an improvement in update performance. Excluding words prevents *noise words* from being indexed as keywords.

The best words to exclude from indexing are words that:

- are meaningless for retrieval (“the”, for example)
- occur too frequently to distinguish records (“Inc.”, for example)

Creating an excluded words file

You should place words you want to exclude from keyword indexing in an ASCII file. This is called an *excluded words file*. A sample excluded words file, EXCLUDES.PUB.DISC, is provided with the software. It contains a list of commonly excluded words, and can be copied and modified to suit your databases. You also can create an excluded words file from scratch if you want.

When you exclude words for No Translate keys, you must be sure to include the different case sensitive spellings of the word. For example, if “CO” is an excluded word, the keyword “Co” will still be indexed for any No Translate keys.

Loading the excluded words list

When you have created an excluded words list to customize the indexing of 8-bit ASCII characters, you must load it using OmniUtil prior to indexing your data. Follow the steps below.

1. Run OmniUtil:
 RUN OMNIUTIL.PUB.DISC
2. From the Main Menu, select 5. Configuration Options.

3. From the Configuration Options submenu, select 1. Database Options. If you have not opened a database at this point, OmniUtil prompts you for one. Enter the name of the database for which you want to exclude words, or press [F2] to select one from a pick list.
4. From the Database Options submenu, select 2. Exclude Words from Indexing.
5. OmniUtil prompts you for the name of the excluded words file. Enter the name of the file. The example below shows the filename "EXCLUDES.PUB.DISC" entered for the excluded words list.



```
Filename (SALESEX.DEMO08): EXCLUDES.PUB.DISC
```

6. OmniUtil tells you that it is copying the file you specified to the file "*dbname*EX", where *dbname* is the name of the current database. For example, if you specified a database named "SALES", the name of the file that OmniUtil creates for that database is named "SALESEX". Select (OK) to proceed.

When it is finished, OmniUtil tells you Excluded word list has been loaded. The file resides in the same group as its database. Select (OK) to proceed.

7. Specifying excluded words does not remove them from the indexes if they are already indexed. After you exclude words from indexing for a database, you must reindex the database to remove any excluded words that were already indexed.

After OmniUtil has loaded the excluded words list you specified in step 5 into the excluded words list for that database (*dbname*EX), it opens the reindexing submenu so that you can rebuild your OMNIDEX indexes. See "Loading the Indexes", on page 2-28, for more information about loading the indexes.

Now, whenever you reindex any OMNIDEX keys for the database, OmniUtil looks for the file "*dbname*EX" in the group where the database resides and uses it to exclude words from the indexes.

To index excluded words for any keyword key, install the No Exclude option on that column. The No Exclude option is discussed on page 2-15.

To remove the excluded words list and index all words, delete the file "*dbname*EX" from the database's group and reindex the keyword keys in your database.

Customizing the Translation of 8-bit Characters

When OMNIDEX indexes 8-bit ASCII characters it translates many of them into a 7-bit equivalent. Others are left untranslated, but can still form part of an OMNIDEX keyword. The remaining characters are treated as delimiters for keywords. You can override this default translation using OmniUtil, as discussed next.

Creating a translation table

The default translation of 8-bit characters is based on the Roman8 character set, and is discussed in greater detail in “How OMNIDEX translates 8-bit characters”, on page B-8.

To alter how OMNIDEX translates characters, create an ASCII text file with 255 lines, one for each 8-bit character in ASCII collating sequence. For each character you want to translate to a different ASCII counterpart, place the desired ASCII character in column 1 of the line that corresponds to the 8-bit character's position in the collating sequence. To leave a character untranslated but able to be part of an OMNIDEX keyword, enter a backslash (\). To use the default translation for that character leave the line blank.

For example, the Greek8 letter “Φ”, which OMNIDEX translates as a Roman “O” (upper case) by default, occupies position 214 in the ASCII collating sequence. To define that character, “Φ”, as a Roman “F”, type the ASCII character F in column 1 of line 214. To translate the lower case “phi” (φ) to “f”, type an f on the appropriate line in collating sequence (here, line 230).

Lines 211 through 230 of the translation table might look like this.

```
.  
.
211
212
213
```

```

214      F
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230      f
.
.

```

A blank line in a translation file means the corresponding character is translated to the default equivalent. A backslash (\) in a line means the corresponding character is not translated at all but it can still be part of an OMNIDEX keyword. Therefore, you cannot redefine a character to be blank or a backslash.

Although some characters may represent the combination of two letters, you can only translate them to one ASCII character. Thus "ß" (the German equivalent of "ss" or "sz") can only be "s" or "z" but not "ss" or "sz".

The first 32 8-bit ASCII characters, which are control codes, can be translated to letters, numbers or special characters. The reverse is not allowed, however. You cannot translate a character to a control code.

We recommend that you do not redefine the special characters used internally by OMNIDEX for Boolean operators, ranges, wildcards, and so on. These characters are:

+ , - : @ ? = \$ & *

Loading the translation table

When you have created a translation table to customize the indexing of 8-bit ASCII characters, you must load it using OmniUtil. Follow these steps.

1. Run OmniUtil:
RUN OMNIUTIL.PUB.DISC
2. From the Main Menu, select 5. Configuration Options.
3. From the Configuration Options submenu, select 1. Database Options. If you have not opened a database at this point, OmniUtil prompts you for one. Enter the name of the database to which you want to apply the translation table, or press [F2] to select one from a pick list.
4. From the Database Options submenu, select 4. Load NLS Translation Table.
5. OmniUtil prompts you for the name of the translation table file. Enter the name of the file. The example below shows the filename "TTABLE" entered for the translation table.



```
Filename (SALESTT.DEMODB): TTABLE
```

6. OmniUtil tells you that it is copying the file you specified to the file "*dbname*TT", where *dbname* is the name of the current database. For example, if you specified a database named "SALES", the name of the translation file that OmniUtil creates for that database is named "SALESTT". Select (OK) to proceed.

If the file is successfully copied, OmniUtil informs you NLS Translation Table has been loaded. The file resides in the same group as its database. Select (OK) to proceed.

7. When OmniUtil is finished copying the translation table you specified in step 5, into the translation table for that database (*dbname*TT), it opens the reindexing submenu so that you can reflect the customized translation in your OMNIDEX indexes. See "Loading the Indexes", on page 2-28, for more information about loading the indexes.

Now, whenever you reindex any OMNIDEX keys for the database, OmniUtil looks for the file "*dbnameTT*" in the group where the database resides, and uses it when indexing ASCII characters.

To disable the translation of 8-bit characters for any column, install the No Translate option on that column. For more information about the No Translate option, see "No Translate", on page 2-15.

To restore the translation table to defaults, delete the file "*dbnameTT*" from the database's group.

Loading the Indexes

NOTE

Skip this section if you created and streamed an installation and indexing job by selecting 3. Create/Stream Installation and Indexing Job at the end of the installation.

After OmniUtil creates the OMNIDEX index structures, the next step is to load them with data. If you created and streamed an installation and indexing job at the end of the installation step, the indexes were loaded automatically.

If you activated an installation immediately after specifying keys, or if you saved and activated an installation, you still must load the indexes. You can do this online or in batch mode.

NOTE

The indexing step requires exclusive access to the index structures.

After you activate an installation, OmniUtil tells you that it has successfully created the indexes, and then offers you these options:

```
1. Populate All Indexes Now
2. Create/Stream Indexing Job
3. Set ID Sort Order
4. Set Index Buffer Size
5. Return to the Menu
```

The last choice above takes you back to the Index Installation submenu. The other choices are discussed in the three sections that follow.

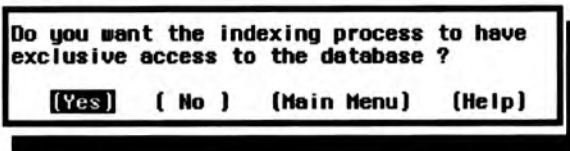
NOTE

If you plan to load a high volume of records into your OMNIDEX database, either load the records before you install OMNIDEX keys, or load the records without updating indexes in real time (see “Batch Updates”, on page 4-52). Index all keys after you have loaded the records.

Indexing in real time

To load all the indexes for a database in real time, choose 1. **Populate All Indexes Now**. OmniUtil displays a dialog box that asks if you want to repopulate all indexes now. Select **(Yes)** to proceed, or **(No)** to stop.

If you select **(Yes)** to index immediately, OmniUtil displays the following dialog box:



OmniUtil can index a database's keys while it is being accessed, as long as its indexes are not being accessed. However, if OMNIDEX keys are being updated during the indexing process, the indexes will not reflect all of those updates. Therefore, to ensure index integrity, select **(Yes)** from the dialog box. OmniUtil then opens the database with exclusive access (mode 8) for indexing.

If you are sure that no OMNIDEX key fields will be updated during the indexing process, or if you don't require complete index integrity, select **(No)** from the dialog box. OmniUtil then opens the database with shared access (mode 5 or 6) for indexing.



If TPI is enabled you must select **(Yes)** at this prompt.

2

Indexing jobs

Generating and streaming an indexing job is another way to load the indexes. To do this, select 2. *Create/Stream Indexing Job* from the menu discussed above. OmniUtil prompts you for the passwords necessary to stream the job. Next, you are asked if you want to stream the job or keep it as a file.

If you choose to keep the file, OmniUtil creates it in your database group and account. The default name is the database name plus the letters "IX". For example, the job file for a database named "SALES" would be "SALESIX".

If you choose to stream the file, OmniUtil prompts you to schedule a time to stream it. Enter any parameters that you would normally include in a *STREAM* command to schedule a batch job. Examples are *AT=* to submit the job at a certain time, and *DAY=* to submit the job on a certain day. If you enter several options, separate them with a semicolon (;). Any options you enter should match the syntax for the *STREAM* command required by the MPE command interpreter.

Index buffer size

To set the buffer size for your index operation, choose 3. *Set Index Buffer Size*. This lets you specify how much memory (in megabytes) to dedicate to the indexing operation. You must enter a value between 2 and 128. See "Tuning indexing performance," in the "Other Performance Considerations" section of the "Topics" chapter, for more information. After you specify a buffer size, OmniUtil lets you proceed with the indexing operation, or create an indexing job file.

Keyword index sorting

The ID Sort feature allows records qualified in an OMNIDEX keyword key retrieval to be returned in order, sorted by a field or fields in the OMNIDEX master data set.



The sort order is established at indexing time, but is not preserved as records are added or deleted from the OMNIDEX domain. To maintain the sort order you must periodically reindex the table and respecify the sort order.

This feature is supported for sets in transparent domains but is not allowed on domains that use double integer search items, or on unlinked detail sets.

To sort by a field during indexing perform the following steps:

1. Select **2. Reindexing Options** from the Main Menu
2. Select **1. Reindex All Indexes** or **2. Reindex Specific Tables** from the Reindexing Options menu. OmniUtil displays a list of tables. Select the table you want to define by highlighting it and pressing **[return]**. Press **[f5]** to continue.
3. Choose **3. Select ID Sort Order** and OmniUtil shows the tables that are eligible for keyword index sorting. Choose a table by highlighting it and pressing **[return]**.
4. OmniUtil shows a list of fields available for sorting. You can choose as many fields as you want, as long as their combined length plus the search item's length does not exceed 248 bytes. OmniUtil sorts by fields in the order they are selected. For example, if you chose to sort PRODUCTS records during indexing, and selected the DATE-MODIFIED field before the SALES-PRICE field, DATE-MODIFIED would be the primary sort field and SALES-PRICE would be the secondary sort field.
5. Press **[f5]** to return to the table menu and **[f8]** to return to the indexing options menu.
6. Select **1. Populate Selected Indexes Now** or **2. Create/Stream Indexing Job**.

Verifying and Testing the Installation

There are several ways to verify how keys are installed on a database.

- Immediately after you run OmniUtil, select 2. View Active Indexes from the Index Installation menu.
- During the installation, press **[f4]** from the Field window.
- At the end of an installation, select 5. View Specified Installation from the activation menu.

The procedures for viewing OMNIDEX keys and their components and options are discussed in “2. View Active Indexes”, on page 3-9.

To test your installation, use DataView to retrieve some records using the keys you installed. There are two ways to run DataView.

- From OmniUtil, access the Main Menu and select 6. Run DataView.
- From an operating system prompt, enter:

```
RUN DATAVIEW.PUB.DISC
```

If DataView prompts you for a database, enter the database name, or press **[f2]** to select from a pick list of databases in your log-on account.

DataView prompts you for a table. Enter the name of a table where you installed keys, or press **[f2]** to select from a pick list of tables in the open database. When you designate a table to query, the DataView Main Menu appears.

Finding and listing records

DataView menus, pick lists, and function keys work the same as OmniUtil menus, pick lists, and function keys. To select an item from a menu or pick list, highlight it using your cursor keys and press **[return]**. To get help, press **[f1]**. To exit from a window to the previous window, or from the Main Menu out of DataView, press **[f8]**. For more detailed information about using DataView, see the “Using DataView” section of the “Utilities” chapter.

Find some records using keyword keys. To do this in DataView select 1. Find Records by Multiple Keys. DataView displays a selection window that contains prompts for all the keyword keys in the current table. Enter some keywords at the search prompts, and watch the number of qualified records, displayed at the bottom of the search window, change to reflect how many records you found.

If you're not sure what words to enter at a search prompt, enter !0:z. DataView displays a pick list of all the keyword values indexed for that key. To enter any of the keyword values, highlight it and press **[return]**.

You can enter several keywords at a search prompt, separated by the Boolean operators AND, OR, and NOT. For a complete discussion of the searches you can perform, see "OMNIDEX Retrievals" on page 4-2 of the "Topics" chapter.

When you are finished entering search arguments, and are ready to list the records you've found, enter /L at any search prompt. DataView displays a pick list of all the records that were qualified by your keyword searches. You can use the cursor keys, and the **[Prev]** and **[Next]** (or **[PgUp]** and **[PgDn]**) keys on your terminal to scroll through the records listed.

To view an individual record from the record list, highlight it and press **[return]**. DataView displays as much of the records as it can on one screen (or "form"). If the record is displayed across several forms, use the **[f3]** and **[f4]** function keys to scroll down and up across forms. To view the next record in the list, press **[f5]**, labeled Next Rec. To view the previous record in the list, press **[f6]**, labeled Prev Rec. You can also use any of the buttons displayed at the top of the forms window (such as (Next) and (First)) to move through the records list.

To find records using sorted keys, select 2. Find Records by Sorted Keys. DataView displays a selection window that contains all the sorted keys in the current table. Select one, and DataView prompts you for a search argument. Remember that sorted search arguments must match the indexed key value, byte for byte, beginning with the leftmost byte. To specify a partial key value, append an @ to the partial argument.

Changing the Installation

There are several ways to change how keys are installed on a database.

- ❑ From the OmniUtil Index Installation menu, select 1. Install or Modify Indexes.
- ❑ During OMNIDEX installation, follow the procedures discussed in “Correcting mistakes” on page 2-16.
- ❑ At the end of an installation, select 4. Modify the Specified Installation from the activation menu.

Note that if you change how keys are installed on a database after you've activated and indexed an installation, you must reactivate those changes and reload the new indexes to reflect those changes.

After you activate the changes to an installation OmniUtil asks you:



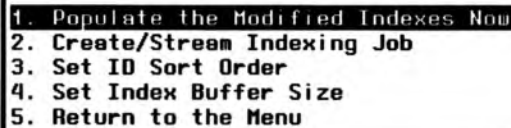
```
Do you want to create a new Installation job
stream for this database?
( Yes )          ( No )
```

Select (Yes) to create an installation job stream that reflects the new OMNIDEX installation. OmniUtil prompts you for the passwords required to stream the job. If there is already an installation job file, OmniUtil asks for permission to overwrite it before it creates the new installation job file.

Reindexing changes

When you change how keys are installed on a database, OmniUtil creates indexes for added keys, or changes existing indexes for keys whose options were changed. You must load (reindex) the indexes to reflect the installation changes you made. Reindexing to incorporate your changes reindexes *all* indexes, not just the ones to which you made changes.

Next, OmniUtil presents you with an indexing menu:

A screenshot of a terminal window showing a menu with five numbered options. The first option is highlighted with a dark background. The menu is enclosed in a double-line border.

```
1. Populate the Modified Indexes Now
2. Create/Stream Indexing Job
3. Set ID Sort Order
4. Set Index Buffer Size
5. Return to the Menu
```

2

Select the appropriate options for indexing your keys. These options are discussed in detail in “Loading the Indexes”, on page 2-28.

Chapter 3: Utilities

This chapter discusses the utility programs that are included in OMNIDEX.

- ❑ “OmniUtil”, on page 3-2, discusses the menu structure and functions of OmniUtil, the index installation and maintenance utility. You can find more information about index installation in chapter 2, “OMNIDEX Installation”. For more information about maintaining OMNIDEX indexes, see “Maintenance”, on page 4-68.
- ❑ “DataView”, on page 3-21, discusses the menu structure and functions of DataView, the database query program that uses OMNIDEX keys to find records. You can find more information about OMNIDEX retrievals in “OMNIDEX Retrievals” on page 4-2 of the “Topics” chapter.
- ❑ “DBMGR”, on page 3-34, discusses the strategies and commands for using the DBMGR database maintenance utility.

OmniUtil

OmniUtil is a user-friendly, menu-driven program that makes OMNIDEX key installation and maintenance fast and easy. You can use OmniUtil to:

- install keyword and sorted keys on a database
- load the OMNIDEX indexes that support keyword and sorted key access
- change the way OMNIDEX keys are installed on a database
- configure OMNIDEX to best suit your needs

All the information you need to use OmniUtil is available through its menus and function keys. Accessing that information is as easy as highlighting a menu item, or pressing a function key.

Program operation

To run OmniUtil, enter the following statement at a system prompt:

```
RUN OMNIUTIL.PUB.DISC
```

The OmniUtil program banner and Main Menu appear, and you can use or inquire about any of the menu items.

To run OmniUtil against a TPI-enabled database installed with OMNIDEX version 3.0 or later, you must run OmniUtil to reference XLOMNIDX.PUB.SYS. You can do this by adding an XL= parameter, as in the run statement below:

```
RUN OMNIUTIL.PUB.DISC;XL="XLOMNIDX.PUB.SYS"
```

Use TPIXLCFG.UTIL to modify the XL path for OmniUtil and the other DISC utilities to direct them to XLOMNIDX.PUB.SYS.

Using the menus

OmniUtil menus are windows that contain numbered lists of items. Some items open another menu window, while other items begin an operation. There are two ways to select from the OmniUtil menus.

- ❑ Use the arrow keys on your terminal or PC to highlight the item, then press **[return]** (or **[enter]** if you are using a PC).
- ❑ Enter the number of the item by typing it and pressing **[return]**.

To exit a menu and return to a previous menu, press **[esc]** twice, or press **[f8]** as described next. From the Main Menu screen, pressing **[esc]** twice, or pressing **[f8]**, exits you from the program.

For information about any menu item, highlight the item and press **[f2]**.

Using dialog boxes

OmniUtil tells you what it's doing by means of dialog boxes. These dialog boxes accept information via buttons. Buttons are words in parentheses, like (OK) and (Cancel). You can select them by highlighting them with the cursor keys and pressing **[return]**.

There are three types of dialog boxes.

The first type of dialog box tells you what your options are for a particular operation. Usually, such dialog boxes contains a (Yes) button to proceed with the operation, a (No) button to cancel the operation, and a (Help) button to provide information about the operation.

The second type of dialog box tells you when OmniUtil is involved in an operation, like installing indexes. Such dialog boxes contain a revolving cursor. When these dialog boxes are on screen, wait for further instructions.

The third type of dialog box tells you when an operation has completed successfully, or when an error occurs. To remove such boxes and proceed, press **[return]**. Some of these boxes contain a (Help) button to provide detailed information about what has occurred.

Prompt windows

Prompt windows contain a blank input field where you can enter such information as the name of a composite key component, or a data set to install with OMNIDEX keys. The cursor must be present in the input field to enter information in it. To enter information, you can do either of two things.

- ❑ Press **[f2]** to get a pick list of appropriate response, like a list of items.
- ❑ Type the information in the input field and press **[return]**.

Often, prompt windows incorporate buttons. For example, after you designate a sorted key in the Field prompt window, you are presented with three such buttons: (OK), (Options), and (Cancel). Selecting (OK) installs the key. Selecting (Options) lets you add indexing options to the key. Selecting (Cancel) rejects the key.

Using the function keys

OmniUtil provides a set of function keys for each screen. The function keys, which correspond to the keys labeled F1 to F8 on your keyboard, change to accommodate the current menu window. For ease of use, however, the most important function keys remain constant. The function keys are listed below by their number, label, and function.

- | | | |
|-------------|---------------------|---|
| [f1] | Help | always displays help about a menu or prompt. The help message displayed depends on which menu or prompt you are at when you press [f1] . Once you are in the help window, you can get more information about the screen by pressing [f3] , or Zoom. You can even get help on using help by pressing [f1] again! To scroll through a help message, use the [Pg Up] and [Pg Dn] (or [Prev] and [Next]) keys on your keyboard, or use the [f6] and [f5] function keys. |
| [f2] | MenuDesc
or Show | When labeled MenuDesc, pressing [f2] displays information about the currently highlighted menu item. When labeled Show, pressing [f2] displays a “pick list” of acceptable answers to a prompt. |

[f3]	Zoom	is labeled only when help is displayed. It lets you view more of the help being displayed.
[f5]	PageDown	is labeled only when help, or a pick list is displayed. Use it to scroll down the length of a list or help screen.
[f6]	Page Up	is labeled only when help, or a pick list is displayed. Use it to scroll up the length of a list or help screen.
[f7]	Refresh	returns the screen to its normal appearance should it become corrupted.
[f8]	Backup Exit or Done	returns you to the previous menu or prompt. From the Main Menu, pressing [f8] exits you from OmniUtil. At an installation prompt, it signals OmniUtil that you are finished installing keys (or tables).

Running in trace mode

You can run OmniUtil in trace mode, which causes it to display the input and output of any internal utility that it calls. Normally, during such operations as activating an installation or loading indexes, OmniUtil displays a window with a revolving cursor. In trace mode, OmniUtil displays the external utilities it calls and what the utility is doing during any operation.

To run OmniUtil in trace mode, enter the following run statement at a system prompt:

```
RUN OMNIUTIL.PUB.DISC;INFO="-T"
```

This feature is especially useful for displaying the number of keywords or key values unloaded, as well as the time required for each step, during indexing. When changing existing installations it lists the OMNIDEX indexes that must be reloaded. It may also be useful for getting more information about internal errors that occur during an installation or indexing operation.

When operations are submitted in batch, OmniUtil is run in trace mode by default.

For more information about using OmniUtil, press **[f1]**, labeled Help, from the Main menu shown in Figure 3-1 on the next page.

The next section describes the OmniUtil menus.

OmniUtil menus

This section describes the OmniUtil menus and discusses each of their options. Note that all the menus share one thing in common: the last item of each menu is `Exit Program`. Selecting `Exit Program` ends your OmniUtil session and returns you to the system prompt.

The Main Menu

This menu appears immediately after you run OmniUtil, and is shown in Figure 3-1. Each of the Main Menu items is discussed next.

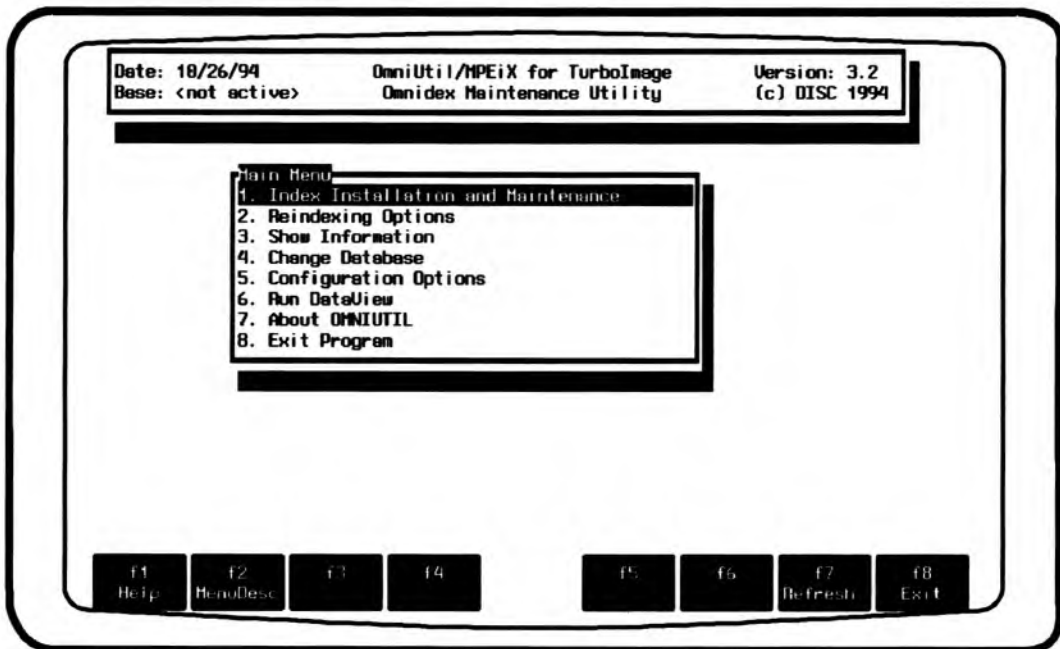


Figure 3-1: The OmniUtil Main Menu

1. Index Installation and Maintenance

This item displays a submenu that lets you install or modify indexes, view active indexes, generate an installation job, load a saved installation, remove all indexes, or convert a previous version's indexes. This item's submenu is discussed in greater detail on page 3-8.

2. Reindexing Options

This item displays a submenu that lets you load the indexes created for OMNIDEX keys for an entire database, for specific tables (data sets), or for specific sorted keys. This item's submenu is discussed in greater detail on page 3-11.

If you have not specified a database, OmniUtil automatically prompts for one whenever you attempt an operation, such as installing indexes, that requires access to one.

3. Show Information

This item provides information about database structure, OMNIDEX indexes, and DISC error messages. Structure provides structure information about databases, tables, and columns. Indexes provides indexing information about databases, tables, and keys only. Messages provides information about OMNIDEX/IMSAM intrinsics, OmniWindow, and OmniUtil messages. When you choose 3. Show Information, OmniUtil prompts you for a database or message string. See page 3-13 for more information about this menu option.

3

4. Change Database

This item lets you change the database to which OmniUtil operations apply. If no database is currently selected, you can use this item to select one. When you choose this item, OmniUtil prompts you for a database name. You can either enter the name of a database, or press **[f2]** to select from a list of databases in your log-on account.

5. Configuration Options

This item lets you configure OMNIDEX for your site's needs. It also lets users change OmniUtil's screen characteristics. When you choose this item, OmniUtil displays a submenu, which is discussed in greater detail in Figure 3-5, on page 3-17.

6. Run DataView

This item runs DataView, the query tool designed for OMNIDEX databases. Use DataView to experiment with OMNIDEX's powerful retrieval capabilities, and to test your key installation. Turn to page 3-21 for more information about DataView.

7. About OmniUtil

This item displays information about OmniUtil's version and the OMNIDEX version installed on your system. This information is useful in resolving technical support issues. This item also lists the telephone numbers of our Response Center.

8. Exit Program

The last item of every menu is `Exit Program`. This item takes you to the following dialog box:



If you select (`Yes`), you return to a system prompt or the last suspended process. If you select (`No`), you are returned to the menu where you selected `Exit Program`.

Index Installation and Maintenance

This submenu corresponds to the first item of the Main Menu. It lets you install OMNIDEX keys and their options on tables (data sets) in your database. The Index Installation and Maintenance submenu is shown in Figure 3-2 on the next page.

While this section discusses the options of the Index Installation and Maintenance submenu, you should read "Installing OMNIDEX Keys" on page 2-3 of the "OMNIDEX Installation" chapter for information about installing and maintaining OMNIDEX keys.

Many of the items on this submenu require access to a database. If you choose such an item before you've specified a database, OmniUtil automatically prompts for one. You can either enter the name of a database, or press **[F2]** to select from a list of databases in your log-on account.

Each of the items in this submenu is discussed next.

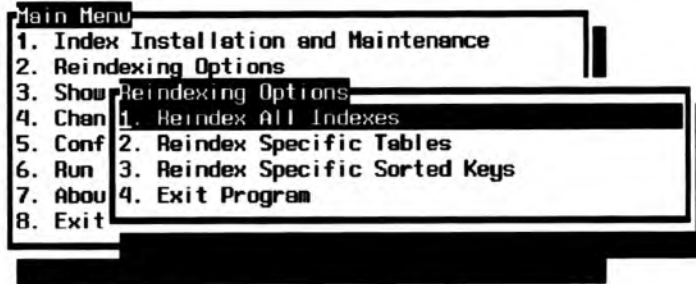


Figure 3-2: The Index Installation and Maintenance submenu

1. Install or Modify Indexes

This item begins the process of installing, changing, or deleting OMNIDEX keys on a database. When you select a database, OmniUtil prompts for a table to install with OMNIDEX keys. Either enter the name of a table, or press **[F2]** to select from a list of tables in the current database. If keys are already installed on the table you specify, OmniUtil displays them in a window.

2. View Active Indexes

This item displays all the tables (data sets) in a specified database that have OMNIDEX keys installed on them. When you specify a database to view, OmniUtil displays a window containing all keyed tables. To view the keys for any given table, highlight it and press **[return]**.

When you specify a table to view, OmniUtil displays a window containing all keys for that table. To view the options installed on any given key, as well as the components of any composite key, highlight the key and press **[return]**.

Press **[F8]** to back out from any window to the previous window. For example, to back out from a list of keys to the list of tables, press **[F8]**.

3. Generate an Installation Job

This item causes OmniUtil to record, in the form of a job file, how OMNIDEX is installed on a database. The installation job that OmniUtil generates is useful for reinstalling OMNIDEX keys, should the need arise. It is also useful for resolving technical support issues. See “Reinstalling OMNIDEX keys” on page 4-70 of the “Topics” chapter for instances where reinstallation is required.

4. Load Saved Installation from File

When you finish specifying OMNIDEX keys for a database, OmniUtil still must create the index structures necessary to support those keys. At the end of an installation session you can choose to save the installation, and create the index structures later.

Typically, you would save an installation if you wanted to create indexes in a database that was being accessed. Having done that, you would use this option to load the saved installation and create the index structures necessary to support it.

5. Remove All Indexes

This item removes all OMNIDEX index structures from a database. To selectively remove only some index structures (and their keys) from a database, select `1. Install or Modify Indexes`, described on page 3-9.

6. Convert Previous Version's Indexes

If you are upgrading from OMNIDEX version 2.02 or later, this option lets you install version 3 indexes that provide the same access (keyword or sorted, plus any key options) that you had under the previous version. When you select this item from the menu, OmniUtil asks whether you want to convert the indexes online or in batch.

As with `1. Install or Modify Indexes` you can load the indexes immediately after installing them, or let OmniUtil create a job stream to load them later.

For a detailed discussion of this menu item, see “Updating an existing OMNIDEX installation”, on page 4-71.

Reindexing options

This submenu lets you load (or reload) the OMNIDEX indexes for an entire database, for specific tables (data sets), or for specific sorted keys. It also provides access to more advanced indexing capabilities. The reindexing Options submenu is shown in Figure 3-3.

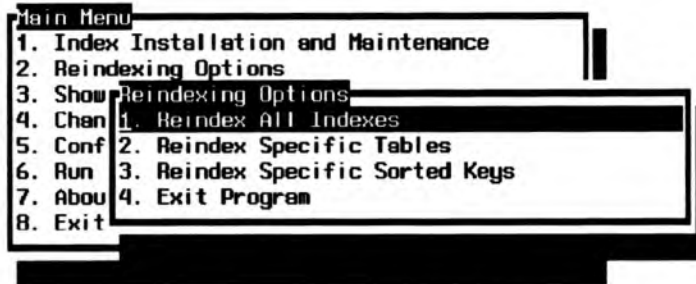


Figure 3-3: The Reindexing Options submenu

3

Many of the items on this submenu require access to a database. If you choose this submenu before you've specified a database, OmniUtil automatically prompts for one. You can either enter the name of a database, or press **[F2]** to select from a list of databases in your log-on account.

Each of the items of the Reindexing Options submenu is discussed next.

1. Reindex All Indexes

This item begins the process of loading (or reloading) OMNIDEX index structures for an entire database. Choosing this item opens a submenu that gives you these options:

1. Populate All Indexes Now loads the index structures immediately and online.
2. Create/Stream Indexing Job creates a job file that loads the index structures, and lets you schedule a time for the job to stream.
3. Set ID Sort Order lets you index records in an OMNIDEX keyword domain in sorted order by specified fields.¹

1. When you select this item from the menu, OmniUtil only displays those sets that are valid for the sorting of OMNIDEX IDs.

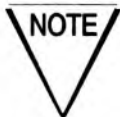
4. Set Index Buffer Size lets you specify how much memory (in megabytes) to dedicate to the indexing operation. You must enter a value between 2 and 128, as discussed in “Other Performance Considerations”, on page 4-97.
5. Return to the Menu returns you to the Reindex All Indexes submenu.

For complete information about loading indexes, see “Loading the Indexes”, on page 2-28.

2. Reindex Specific Tables

This item begins the process of loading (or reloading) OMNIDEX index structures that correspond to all keys in the tables you select. When you choose this item, OmniUtil displays a pick list of tables in the current database. You can select or de-select individual tables by highlighting them and pressing [f2] or [return]. You can select all tables by pressing [f3]. You can deselect all tables by pressing [f4].

When you have selected all the tables you want to reindex, press [f5] to begin reindexing them. Pressing [f5] opens a submenu. See “1. Reindex All Indexes”, on page 3-11, for a picture of this submenu and a discussion of its options.



When several tables are linked in the same domain, selecting any one of them for reindexing reloads the keyword indexes for all of them.

3. Reindex Specific Sorted Keys

Choosing this item begins the process of loading (or reloading) OMNIDEX index structures that correspond to only the sorted keys that you select. When you choose this item, OmniUtil displays a pick list of sorted keys in that database. You can select or de-select individual keys by highlighting them and pressing [f2] or [return]. You can select all keys by pressing [f3]. You can de-select all keys by pressing [f4].

After selecting all the keys you want to reindex, press [f5] to begin reindexing them. Pressing [f5] opens a submenu. See “1. Reindex All Indexes”, on page 3-11, for a picture of this submenu and a discussion of its options.

Show Information

This submenu contains three options:

1. Structure displays structural information about a database, table (data set) or column (field).
2. Indexes displays information about indexes (or keys) installed on a database, table (data set) or column (field).
3. Messages displays information about OMNIDEX/IMSAM API, or OmniWindow error messages.

The Show Information submenu is shown in Figure 3-4. Each option is discussed next in greater detail.

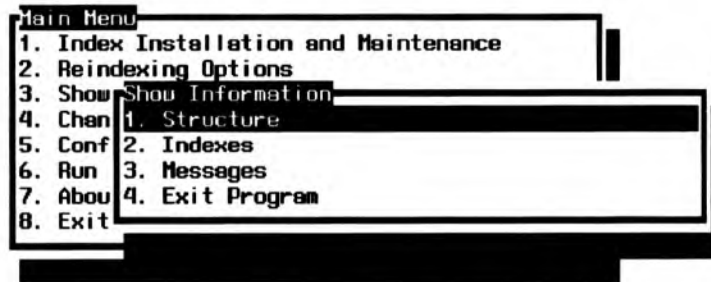


Figure 3-4: The Show Information submenu

1. Structure

This option lets you view structural information about a database. When you choose 1. Structure, OmniUtil displays three options, each of which corresponds to a level of data organization.

1. Base lets you view information about a database.
2. Tables lets you view information about a table (data set).
3. Column lets you view the tables that use the column (item) you specify.

These three options are discussed next in more detail.

1. Base

When you choose this option, OmniUtil prompts you for a database (if one is not already open). When a database is open, OmniUtil displays a dialog box that tells you how many tables and items the database contains.

If you select (*OK*) from the dialog box, OmniUtil lists all tables (data sets) for that database in a pick list format. For each table, OmniUtil lists its set number, type (master or detail), the number of columns it contains, its capacity, the number of records it contains, its length in bytes, and how full it is. You can choose any table to see more information about it, as discussed next.

2. Tables

When you choose this option, OmniUtil prompts you for a table. Pressing **[F2]** displays a pick list of the tables in the open database. When you enter a table, or select one from a pick list, OmniUtil displays the columns (items) that the table comprises. Each column's number, data type, record offset, and size (in bytes) is displayed in a pick list format.

You can select an individual column from the pick list to see more information about it, as discussed next.

3. Column

When you choose this option, OmniUtil prompts you for a column (item). Pressing **[F2]** displays a pick list of the columns in the open database, listed by item number and name. When you enter a column, or select one from a pick list, OmniUtil displays a list of tables that use the column.

2. Indexes

This option lets you view information about how OMNIDEX is installed on a database. When you choose 2. *Indexes*, OmniUtil displays three options, each of which corresponds to a level of data organization.

- | | |
|--------------|--|
| 1. Base | lets you view OMNIDEX key information for a database. |
| 2. Tables | lets you view OMNIDEX key information for a table (data set). |
| 3. Keys Only | lets you view the type of access and options installed on the key you specify. |

These three options are discussed next in more detail.

1. Base

When you choose this option, OmniUtil prompts you for a database (if one is not already open). When a database is open, OmniUtil displays a dialog box that displays the database's name, the date of OMNIDEX installation, the index version, and any indexing products installed (OMNIDEX or IMSAM).

If you select (**OK**) from the dialog box, OmniUtil lists all tables (data sets) for that database in a pick list format. For each table, OmniUtil lists its set number, the number of columns it contains, the number of records it contains, and the types of keys (keyword and sorted) installed on it. You can choose any table to see more information about it, as discussed next.

2. Tables

When you choose this option, OmniUtil prompts you for a table. Pressing **[f2]** displays a pick list of the tables in the open database. When you enter a table, or select one from a pick list, OmniUtil displays the columns (items) that the table comprises, as well as each column's number, data type, whether it's a keyword key, its group number (if it belongs to a group), and whether it's a sorted key.

You can select an individual column from the pick list to see more information about it, as discussed next.

3. Keys Only

When you choose this option, OmniUtil prompts you for a table. Pressing **[f2]** displays a pick list of the tables in the open database. When you enter a table, or select one from a pick list, OmniUtil displays the columns (items) that the table comprises, as well as each column's number, data type, whether it's a keyword key, its group number (if it belongs to a group), whether it's a sorted key, and whether it's a unique key (an SI).

When you select a key from the key pick list, OmniUtil displays the types of access and key options installed on it.

3. Messages

This item provides information about error messages returned by OMNIDEX, OmniWindow and OmniUtil. When you choose this option, OmniUtil displays three options, each of which corresponds to a DISC software product.

1. OMNIDEX/IMSAM API provides information about IMSAM and OMNIDEX error messages.
2. OmniWindow provides information about OmniWindow error messages.
3. OmniUtil provides information about OmniUtil error messages.

When you choose any of the three options listed above, OmniUtil prompts you to enter a message keyword. You can enter a word from the message, or a message number, to obtain a list of messages.

For example, if you select 1. OMNIDEX/IMSAM API, and enter the error number -212, OmniUtil displays the following list of error messages:

```
IMS-212  No current key
ODX-212  List must contain a generic keyword or
         keyword range
```

You can highlight any message from the list and press **[return]**. OmniUtil displays a dialog box for the message, which lets you find more information about the error message. Selecting the (Help) button from the message's dialog box opens a window with information about what could cause the error, and how to remedy or prevent the error.

Configuration Options

This submenu lets you configure OMNIDEX for your site's needs. It also lets users change OmniUtil's screen characteristics. The Configuration Options submenu is shown in Figure 3-5.

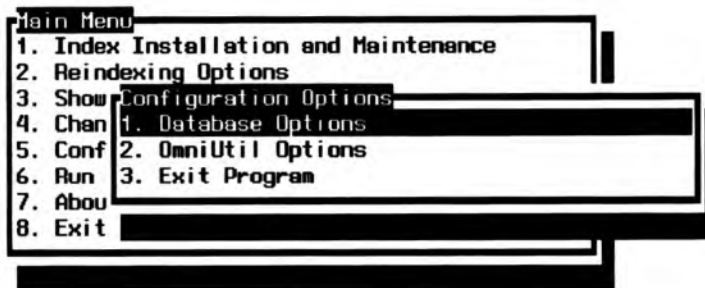


Figure 3-5: The Configuration Options submenu

3

1. Database Options

This item displays a submenu of three items that you can configure for a given database. If you have not specified a database, OmniUtil automatically prompts for one when you choose this item. Each configurable item is listed below with a description of what it affects, and a reference to more detailed information.

1. Configure TurboIMAGE/XL Direct Interface

This option lets you enable a database for Third Party Indexing through the enhanced TurboIMAGE intrinsics that reside in XL.PUB.SYS.



This option only applies to systems using TurboIMAGE version C.04.03 or later or the MPE/iX operating system (version 4.0 or later).

When you select this item, OmniUtil displays a dialog box that shows the current TPI status and prompts for a new status.

```

The TurboIMAGE/XL Direct Interface (TPI) is
currently OFF. Please select new setting:

[ On ]           [ Off ]
  
```

If you select (On) the OMNIDEX indexes are recognized by the TurboIMAGE/iX update intrinsics. Therefore, indexes are automatically updated when you update your data through TurboIMAGE/iX.



Please run the TPIXLCFG.UTIL.DISC command file if you are accessing TPI-enabled databases.

If you select (Off) the OMNIDEX indexes are transparent to the TurboIMAGE update intrinsics and are not automatically updated, unless you run your update programs through Call Conversion (XL.PUB.DISC).

2. Exclude Words from Indexing

This lets you exclude certain keywords from being indexed for a given database. Words like “the”, “and” and “it” are typically useless in keyword searches, take up disk space, and increase the time it takes to load keyword indexes. Excluding keywords is discussed in “Excluding Words from Indexing” on page 2-22 of the “OMNIDEX Installation” chapter.

3. Set ID Sort Order

This lets you sort records in keyword indexes by sort fields that you designate before you begin an indexing operation. The entire process for establishing a sort order is outlined in “Keyword index sorting”, on page 2-31.

4. Load NLS Translation Table

This lets you load a translation table to override the default translation of 8-bit ASCII character sets to a default, 7-bit ASCII equivalent. To override the default translation, you must create a translation table. The entire process for overriding the default translation of 8-bit ASCII characters is discussed in “Customizing the Translation of 8-bit Characters”, on page 2-24.

2. OmniUtil Options

This item displays a submenu of three items that let you configure the look of the OmniUtil menu screens. Each configurable item is listed below with a description of what it affects.

1. Color Configuration

This item lets you change the way the OmniUtil windows look to you. On standard monochrome HP terminals, you can configure blinking, underlined, or reverse highlighted characters. If you run OmniUtil from a PC with a color monitor, you can configure screen colors.

When you choose this item, OmniUtil displays a list of the windows that you can configure to the left of the screen. Sample windows are displayed across the top of your screen to show you how your configuration will look when activated. Both the window list and sample windows are shown in Figure 3-6.

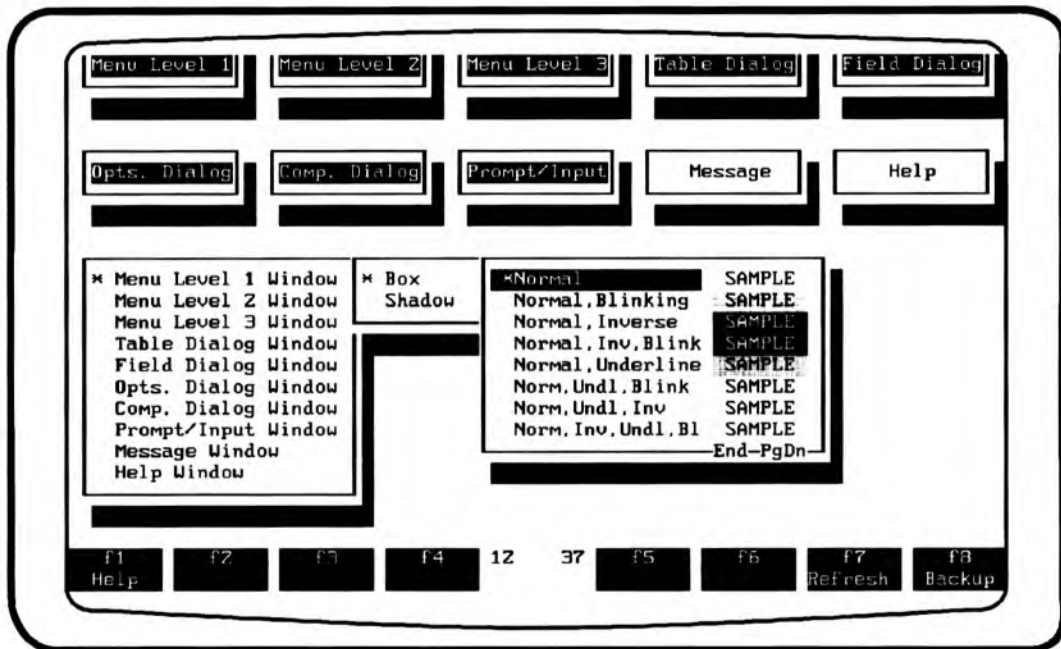


Figure 3-6: The Color Configuration screen

For each window, you can configure the “box” (the actual window that contains the window text), or the “shadow” (the drop shadow behind the window). To configure any window, simply select it from the list. Next, select either the Box or Shadow option. A list of sample configurations appears to the right of the screen.

When you select one of these configurations, you are returned to the `Box` or `Shadow` menu. When you exit the `Box` or `Shadow` menu and return to the window selection menu, the corresponding sample window at the top of the screen reflects your selected configuration.

After you have configured as many windows as you wish, exit the window selection menu. When you do, OmniUtil asks if you want to save the color configuration to a file. If you select `(Yes)`, OmniUtil prompts you for the name of a file. You can either supply the name for the configuration file or press `[return]` to assign the default name "USERCFG". The file is created in your log-on group.

If you assign the default name, "USERCFG", OmniUtil automatically configures its windows based on information in that file whenever you log on as the creator of USERCFG. If you assign a different name, you must use `2. Activate Color Configuration` to load the configuration information stored in that file.

2. Activate Color Configuration

This item lets you load a set of configurations from a file other than the USERCFG that you created using the `Color Configuration` menu, discussed above. This file can be a USERCFG that someone else created, or a configuration file that is not named USERCFG. When you choose this item, OmniUtil prompts you for the name of a file. You can either supply the name of a configuration file or press `[return]` to use the default configuration file (USERCFG).

You can use this option to load files for more than one set of configurations. This is useful if you alternately access OmniUtil from a monochrome terminal, as well as a PC with a color monitor.

DataView

This section discusses DataView, the query utility that provides instant online access to records via OMNIDEX keys. DataView uses some of the features of OmniWindow including:

- ❑ easy to use retrieval windows
- ❑ a pick list of qualified records
- ❑ extensive online help

If you enjoy the easy-to-use features of DataView, you might consider using OmniWindow to develop OMNIDEX applications. OmniWindow can provide all of the features of DataView, and more.

3

Program operation

Running DataView

You can run DataView from an operating system prompt, or from the OmniUtil Main Menu. To run DataView from the operating system, enter:

```
RUN DATAVIEW.PUB.DISC
```

To run DataView from the OmniUtil Main Menu, select 6. Run DataView. If you have already opened a database in OmniUtil, DataView queries that database.

When you run DataView, if no database is currently open, it prompts you for one. You can either enter a database name, or press **[f2]** to select from a pick list of databases in your log-on account. To select an item from a pick list, use the arrow keys on your terminal or PC to highlight the item, then press **[return]** (or **[enter]** if you are using a PC). If you press **[return]** without supplying a database name DataView terminates and you return to the operating system or OmniUtil.

When you supply the name of a database, DataView asks for a password. When you supply a password, DataView returns the message *Please wait*, opening the database.

When the database is open, DataView prompts you for a table (data set). This is the table that you can search for records. You can either enter a data set name, or press [F2] to select from a pick list of tables that are available to your user class. The table that you select must have OMNIDEX keys installed on it to support searches using DataView.

When you supply a database and table name, DataView displays their names in the program banner at the top of the screen. Then the Main Menu appears as shown in Figure 3-7. You can use or inquire about any of the menu items, as discussed on page 3-24.

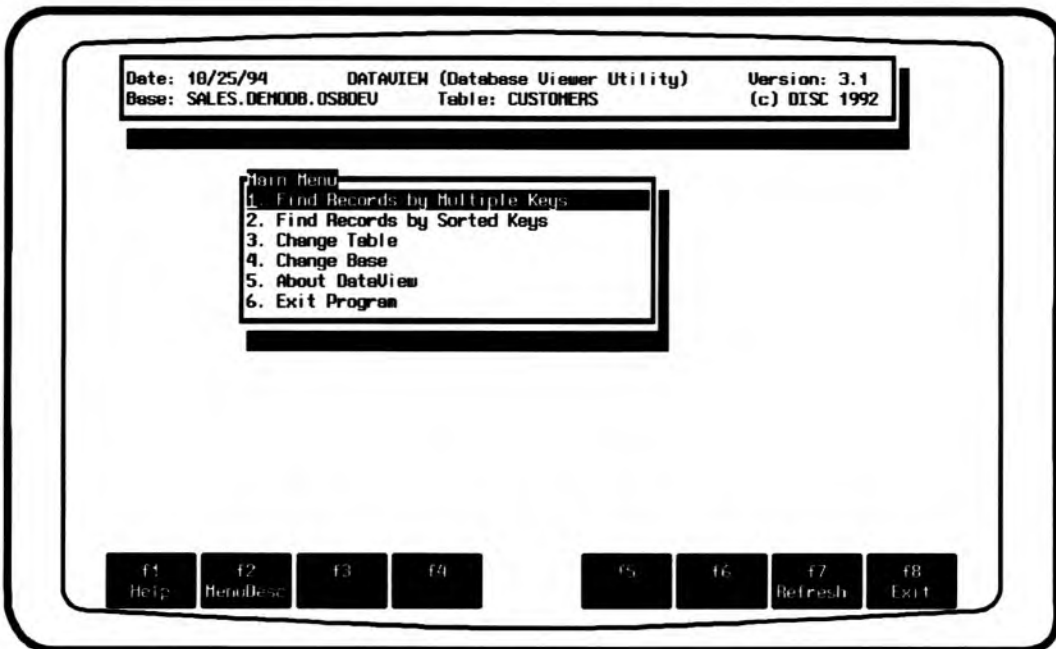


Figure 3-7: DataView's Main Menu

Using DataView windows

Note that most of DataView's windows let you use the cursor keys on your terminal. The only window that does not support the cursor keys on your terminal is the selection window, which is discussed under "Finding records", on page 3-26.

To exit any window (except for the selection window), press **[esc]** twice, or press **[f8]**. This returns you to the previous window. From the Main Menu screen, pressing **[esc]** twice, or pressing **[f8]**, exits you from the program to the operating system or OmniUtil.

Using the function keys

DataView uses function keys as short cuts to entering information, and to access its extensive online help. The function keys, which correspond to the keys labeled F1 to F8 on your keyboard, change to accommodate the current menu window. For ease of use, the most important function keys remain constant. The function keys are listed below by their number, label, and function.

[f1]	Help	always displays help about a menu or prompt. The help message displayed depends on which menu or prompt you are at when you press [f1] . To scroll through a help message, use the keys on your keyboard labeled Page Up and Page Down, or use the [f6] and [f5] function keys.
[f2]	MenuDesc or Show	When labeled MenuDesc, pressing [f2] displays information about the currently highlighted menu item. When labeled Show, pressing [f2] displays a pick list of acceptable answers to a prompt.
[f7]	Refresh	refreshes the DataView screen should it become corrupted.
[f8]	Backup Exit or Done	returns you to the previous menu or prompt. From the Main Menu, pressing [f8] exits you from DataView.

The function keys **[f3]** through **[f6]** are different for each window. Therefore, they are discussed in greater detail as each window is discussed.

The Main Menu

The DataView Main Menu contains a numbered list of items. There are two ways to select an item.

- ❑ Use the arrow keys on your terminal or PC to highlight the item, then press **[return]** (or **[enter]** if you are using a PC).
- ❑ Enter the number of the item by typing it and pressing **[return]**.

For information about any menu item, highlight the item and press **[f2]**.

The DataView Main Menu screen offers the options listed below:

1. Find Records by Multiple key
2. Find Records by Sorted key
3. Change Table
4. Change Base
5. About DataView
6. Exit Program

Each of these options is discussed below under its own heading.

1. Find Records by Multiple keys

When you select this item, DataView lets you search keyword keys installed on the current table (displayed in the program banner at the top of the screen). This is the table that you specified right after you ran DataView, or by selecting **3. Change Table** from the Main Menu.

DataView opens a record selection window that prompts you for search arguments, as shown in in Figure 3-8, on page 3-27. The actual record selection process is discussed in “Selecting records using keyword keys”, on page 3-27.

2. Find Records by Sorted keys

When you select this item, DataView lets you search the current table using sorted keys. DataView displays a pick list of the sorted keys available for the current table. When you choose a sorted key from the pick list, DataView prompts you for an argument. The actual record selection process is discussed in “Selecting records using sorted keys”, on page 3-31.

3. Change Table

This item lets you begin or continue a search in another table within the open database. When you select 3. Change Table, DataView prompts you for a table. Either enter the name of the table, or press [F2] to select from a pick list of tables available to your user class.

When you specify a table, DataView returns you to the Main Menu and displays the table's name in the program banner at the top of the screen.

4. Change Base

This item lets you close the current database and open a new one. When you select 4. Change Base, DataView prompts you for a database. Either enter the name of the database, or press [F2] to select from a pick list of databases in your log on account.

When the database is open, DataView prompts you for a table (data set). This is the table that you can search for records. You can either enter a table name, or press [F2] to select from a pick list of tables that are available to your user class.

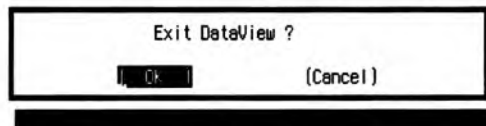
When you specify a database, DataView returns you to the Main Menu and displays the database's name in the program banner at the top of the screen.

5. About DataView

This item provides information about the version of DataView, and the OMNIDEX intrinsics installed on your system. It also lists customer support hours and telephone numbers.

6. Exit Program

This item terminates DataView and returns you to the operating system or last suspended process. When you select 6. Exit Program, a fail safe prompt appears that asks if you want to exit DataView.



If you select (OK) from the fail-safe menu, DataView terminates. If you select (Cancel), you return to the Main Menu.

Finding records

DataView's main function is to let you find records using OMNIDEX keys. When you select 1. Find Records by Multiple Keys from the Main Menu, DataView opens a selection window that lets you find and list records using keyword keys. When you select 2. Find Records by Sorted Keys from the Main Menu, DataView opens a selection window that lets you find and view records using sorted keys. Each of these options is discussed below.

Maneuvering through prompts

There are a variety of function keys and tokens that you can use to move among the prompts in any selection window. This section discusses the function keys and selection tokens that you can use to maneuver through search prompts.

The **[return]** key is the most important key in maneuvering through prompts. You must press **[return]** after you enter search arguments to begin a search. Pressing **[return]** at a search prompt without entering arguments takes you to the next prompt.

In addition to the **[return]** key, there are several tokens you can enter at a search prompt to perform a variety of functions. They are listed below according to function.

If you need to...	Enter...
back up to the previous prompt	^
configure window colors	/C
exit the selection screen with no value returned	^^ or /E
display help for the current prompt	?
display help with prompt command tokens	??
list the qualified records	/ or /L
display a pick list of available commands	// or /M
reload the previously qualified list of records	*
restart the prompting sequence and reset the record list	<< or /S

If you need to...	Enter...
evoke the Task Manager or specific task	/# [taskname]
transfer the current list immediately to a file named <i>file</i>	}file
transfer the current list to a file after answering questions from a dialog box	/X
get version and other global information	/V

Selecting records using keyword keys

To find records from the current table using a keyword key, choose 1. Find Records by Multiple Keys from the Main Menu. Finding records using keyword searches involves entering search arguments at prompts that DataView configures for each keyword key in the current table. DataView supports all of the keyword arguments and search operations described for keyword keys in “Keyword searches” on page 4-2 of the “Topics” chapter.

If the current table has keyword keys installed, a record selection screen appears. This screen contains one prompt for each keyword key or group. For each key, DataView uses its item name, or the name assigned to it during installation (for composite keys), as the prompt string. For a group of keys, DataView uses the name of the first key installed in the group as the prompt string.



Figure 3-8: DataView keyword search prompts

When you enter a keyword argument at any keyword prompt, DataView returns a qualifying count at the bottom of the search window. This count represents the number of records that qualified for the arguments entered. In Figure 3-8, the argument DATA AND SERVICES was entered against the CUSTOMER-NAME key of the CUSTOMERS master table.

As you can see from the `Qualified records: message`, only 10 CUSTOMERS records have both the keywords “data” and “services” in their CUSTOMER-NAME field.

The qualifying count, which follows the `Qualified records: message`, almost always reflects the number of individual records that qualified in a search. The only exception to this rule is when you search keyword keys in detail tables that have the Record Complex option installed on them. Searches on Record Complex keys reflect the number of master records whose detail chains qualified for the arguments entered.

To perform a keyword-only search, precede a range or generic argument with an exclamation point (!). For example, if you’re not sure what words to enter at a search prompt, enter `!0:z`. DataView displays a pick list of all the keyword values indexed for that key or group. To enter a value from the pick list, highlight it and press `[return]`.

After you finish searching, you can list the qualified records by entering `/L` at a search prompt, or by pressing `[f5]`. You can then select and view individual records from that list. This is discussed in “Listing and viewing records”, on page 3-32.

Searching across tables

It is possible to find records in a master table, and then find individual detail records that belong to those master records. You can even qualify the detail record list with additional keyword searches. For example, if you wanted to find all the customer records for the state of California, you would search the CUSTOMERS table by entering `CA` at the prompt that corresponds to the STATE keyword key.

If you wanted to see how many of those customers showed any sales activity since January of 1992, you would have to search the DATE-ENTERED key of CUSTOMER-NOTES with the argument `>9201@`.

Searching across tables involves these basic steps.

1. Find records in the first table.
2. Change to a linked table through the DataView Main Menu.
3. Reload the records (internal ID list) you qualified in the keyword search of the first table.
4. Further qualify the associated records in the second table.

The first two steps are very straight forward. Just use keyword arguments to qualify records in the first table. Then, access the Main Menu, select 3. Change Table and enter the name of a table in the same OMNIDEX domain. See “Linking Details to Masters” on page 4-56 of the “Topics” chapter for more information about OMNIDEX domains.

To load the record list into the selection window for the second table, select 1. Find Records by Multiple Keys from the Main Menu. Then, enter an asterisk (*) at any search prompt. You can tell when records have been qualified by the appearance of a qualifying count. At this point, you can further qualify the records in the second table by entering arguments at the appropriate search prompts.

If you begin a search in a master table and continue it in a detail table, the outcome of the search depends on whether or not the keys in the detail are installed with the Record Complex option. If the keys you search in the detail are not Record Complex keys, then you qualify individual detail records. If the keys you search in the detail are Record Complex keys, then you qualify record complexes. Record complexes are discussed in “Linking Details to Masters” on page 4-56 of the “Topics” chapter.

Whether you qualify individual detail records or entire record complexes affects how you list the qualified records and view individual records, as discussed in “Listing and viewing records”, on page 3-32. When you search for records using Record Complex keys, DataView qualifies master records and their record complexes. Then, when you list or view records, DataView displays master records, even though the current table is a detail. To view records from the detail table you searched, highlight the (Detail) button, displayed above the record window, as discussed on page 3-33.

Transferring qualified record IDs to a file

After you qualify records in a search, you can list them in their internal format to a file for use in future searches. There are two ways to do this.

- Enter }*filename* at any search prompt, where *filename* is the name (which you can qualify to the group level) you're assigning to the temporary file that will contain the internal (record) ID list.
- Enter /X at any search prompt and answer the prompts that DataView displays.

If you enter *filename*, the internal ID file you create is temporary. You can save it as a permanent file using the MPE SAVE command. The easiest way to save the internal ID list as a permanent file, is to enter */X*.

When you enter */X*, DataView first prompts you for a transfer file name. You can enter any legal MPE file name, and qualify the file's name to the group level.

Then, DataView lets you create the file as temporary or permanent. Select one or the other by highlighting it and pressing **[return]**. Then, DataView asks what type of file you want to create.

INTERNAL	creates a new file to contain the internal IDs of the records qualified in the most recent search.
OVERWRITE ASCII	overwrites the contents of an existing file with the internal IDs of the records qualified in the most recent search.
APPEND ASCII	adds the internal IDs of the records qualified in the most recent search to an existing file. The file to which you add the IDs must be an internal ID file, or DataView returns the message: Unable to transfer the records. Status = -801.

Highlight any of the three choices and press **[return]**.

If the transfer is successful, DataView tells you how many records it transferred. For more information about the intrinsic used to transfer data, see "ODXTRANSFER" in chapter 2 of the *OMNIDEX ImagePlus SDK API Guide*.

You can use an internal ID file to search the domain where the IDs were qualified. If you use the files to search a domain other than the one where the records were qualified, DataView uses the IDs, but they do not return the appropriate records. In fact, they may not correspond to any records, and may cause DataView to issue an error message and abort.

To use internal ID files in a search, enter the file's name preceded by a dollar sign (\$) at a search prompt. For example, to use a file named "RECORDS," enter \$RECORDS at a search prompt for any table in the appropriate domain.

Selecting records using sorted keys

To select records from the current table using a sorted key, choose 2. Find Records by Sorted Keys from the Main Menu. When you do, DataView displays a pick list of the sorted keys available for that table. Select a sorted key from the pick list, and DataView displays a search prompt for that key.

When searching sorted keys, remember that you must supply an argument that matches the stored key value byte for byte. When using partial-key arguments, use an at-sign (@) to indicate that the argument is partially specified. Note that partial key arguments are only supported for type U or X keys.

If a sorted key search does not qualify any records, DataView displays a message that says `No records qualify`. When a sorted key search qualifies records, DataView automatically displays the first record. You can view each record in the chain, as described in "Listing and viewing records", on page 3-32. When you reach the end of the chain of records, DataView tells you so.

3

Displaying the number of records found

You can display the number of records found in a sorted key search by pressing **[F2]** labeled `Count On`, before you enter a search argument. When the record count is enabled, **[F2]**'s label changes to `Count Off`. Then, when you qualify records, the forms display (shown in Figure 3-9) tells you which record you are viewing and the length of the chain (for example `Record 2 of 8`) in the lower left corner of the form window.

When count is off, **[F2]**'s label changes to `Count On`. The forms display only tells you which record you are viewing (for example, `Record 2`).

Listing and viewing records

To list the currently qualified records from a keyword search, enter /L at a search prompt. When you do, DataView displays a pick list of qualified records. To view any one of these records, use your cursor keys to highlight it and press **[return]**.

When you select a record for viewing from a pick list, DataView displays it. When a sorted key search qualifies records, DataView automatically displays the first record. When records are too big to fit on one screen, they are displayed across several “forms”. The number of forms that a record comprises is displayed at the bottom of the record window. How to scroll across forms is discussed next.

DataView provides a set of function keys and buttons (such as **(Next)** and **(Prev)** shown in Figure 3-9) that let you move through the qualified records.

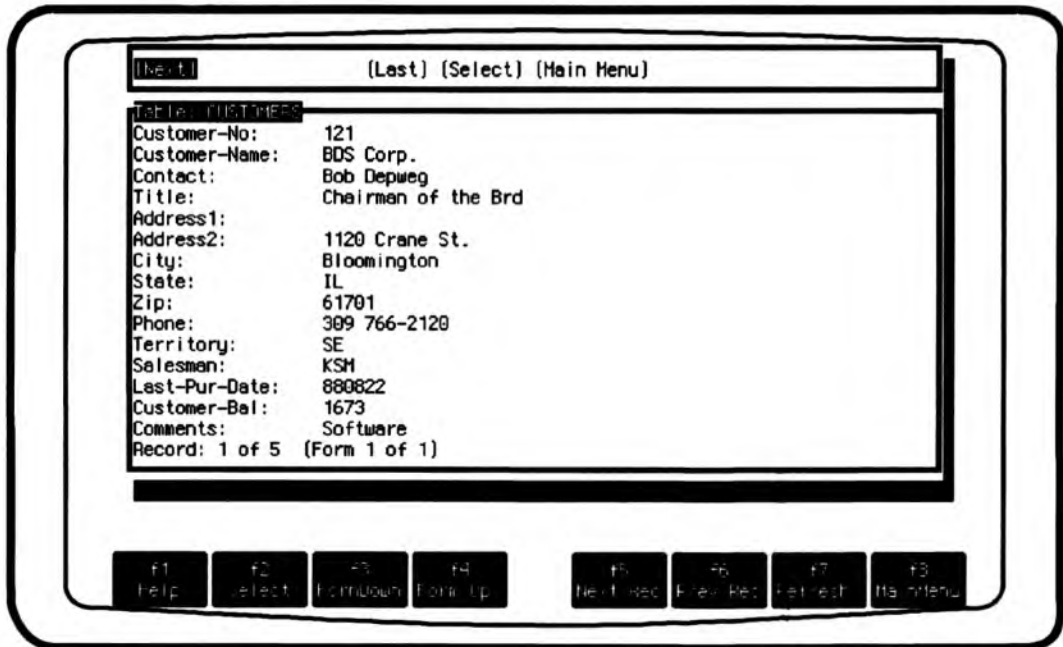


Figure 3-9: Viewing individual records in DataView

Function keys **[f1]**, **[f7]** and **[f8]** act as described on page 3-23. The rest of the function keys, their labels and their functions are listed below.

[f2]	Select	returns you to the selection screen and lets you perform another search.
[f3]	Form Down	scrolls down a form for records that are too big to be displayed in one form.
[f4]	Form Up	scrolls up a form for records that are too big to be displayed in one form.
[f5]	Next Rec	displays the next record in the list of qualified records.
[f6]	Prev Rec	displays the previous record in the list of qualified records.

In addition to the function keys discussed above, DataView displays several buttons at the top of the record list window. You can use these buttons by highlighting them and pressing **[return]**. They are listed below in order of appearance.

(Next)	displays the next record in the list of qualified records.
(Prev)	displays the previous record in the list of records.
(First)	displays the first record in the list of qualified records.
(Last)	displays the last record in the list of qualified records.
(Select)	returns you to the record selection screen.
(Main Menu)	returns you to the Main Menu.
(Detail) or (Master)	When a master record is displayed after a keyword search, selecting (Detail) displays the first detail in the table where the keyword search was performed. When a detail record is displayed after a keyword search, selecting (Master) displays that record's OMNIDEX master record.

DBMGR

DBMGR is DISC's database support tool. It lets you:

- manage data set capacities
- copy and rename data sets
- erase and reload data sets
- delete data sets and data items



Do not use DBMGR on “jumbo” data sets or data sets that have DDX (Dynamic Detail eXpansion) enabled.

Program operation

To run DBMGR, sign on to the account and group where the database you are maintaining resides:

```
HELLO MGR.account,group
```

Then, run the program:

```
RUN DBMGR.PUB.DISC
```

When you run DBMGR, it prompts you for a database name. This is the database to which any maintenance operations will apply.



You must have exclusive access to the database you specify at the database name prompt. If you do not, DBMGR displays a `Database in use error message`.

To determine who is using a database, make sure you're signed on to the account and the group where the database resides. Then, enter:

```
RUN DBUTIL.PUB.SYS  
SHOW dbname USERS
```

Dbname is the name of the database you want to access.

Timing estimates for DBMGR operations

The following formulas and examples are estimates of the time DBMGR takes to perform specific operations. These estimates are based on TurboIMAGE/3000, an HP 3000 922LX, an unloaded CPU and a record size of 128 words (one sector). Actual times for your system and databases may vary, due to differing CPUs and disk drive configurations.

CAP(acity)

The CAP command is used to change the capacity of a data set. The time required to resize each type of TurboIMAGE data set is:

- Master data sets:** 9,000 entries per minute
- Detail data sets:** 80,000 sectors per minute

For example:

- ❑ CAPacity performed on a master data set that contains 90,000 entries = **ten minutes**
- ❑ CAPacity performed on a detail data set that occupies 160,000 sectors = **two minutes**

RELoad

The RELoad command is used to reload detail sets. The formula for determining how long a RELOAD operation should take is:

$$\frac{\text{details} \times \text{paths}}{4000} = \text{minutes}$$

where *details* is the number of detail records, and *paths* is the number of paths to masters.

For example:

- ❑ A RELoad performed on a detail data set with 60,000 records and one (1) path = **15 minutes**
- ❑ A RELOAD performed on a detail data set with 60,000 records and three (3) paths = **45 minutes**

Note that these estimates apply only to unsorted TurboIMAGE paths where the average chain length per SI is eight or more. Reloading details takes more time when they use sorted paths are or when they average less than eight records per chain.

ERASE

The ERASE command is used to remove all entries from a set, or a master set and all details with paths to it. The times required to erase individual sets are listed below.

Manual masters:	70,000 sectors per minute
Stand-alone details:	70,000 sectors per minute
Detail linked to masters: (in seconds)	$\frac{det + auto1 + auto2 + \dots + (2(man1 + man2 + \dots))}{1000}$

The variables shown in the expression above are defined below.

<i>det</i>	is the number of sectors occupied by the detail.
<i>auto1</i>	is the number of sectors occupied by the first auto master linked to it.
<i>auto2+...</i>	is the number of sectors occupied by all other auto masters linked to it.
<i>man1</i>	is the number of sectors occupied by the first manual master linked to it.
<i>man2+...</i>	is the number of sectors occupied by all other manual masters linked to it.

For example:

- ❑ An ERASE performed on a 70,000-sector master = **one minute**
- ❑ An ERASE performed on a 120,000-sector detail linked to two (2) manual masters, each set occupying 60,000 sectors = **six minutes**.

These estimates assume that the average chain length per SI is eight records or more. Erasures of detail sets with paths to masters takes longer when chain lengths are less than eight records.

The erasure time for detail sets that are linked to multi-path auto masters varies based upon the contents of the multi-path auto masters and other detail sets linked to them. Such detail sets typically erase more slowly than details linked only to manual masters and single-path auto masters.

COPY

The COPY command copies a database and any OMNIDEX indexes associated with it at the rate of 80,000 sectors per minute.

For example:

- ❑ A COPY performed on a database whose data sets and indexes occupy a total of 400,000 sectors = **five minutes**.

DBMGR and security

To use the DBMGR CAPacity, ERAse, PRImary, RELoad, DElete and REName commands, you must be logged on as the database creator. You must have read, lock and write access to the database.

The database creator is often the manager of the account (MGR.*account*). To determine the identity of the database creator, you must be logged on as account manager for the account where the database resides. At an operating system prompt, enter LISTF *dbname*,**3** where *dbname* is the name of the database you want to access.

You need not be the database creator to use the COPY or FOrM commands. However, the following security restrictions apply.

- ❑ **If you have system manager (SM) capability**, you can use the COPY or FOrM commands on any database on the system.
- ❑ **If you have account manager (AM) capability**, you can use the COPY or FOrM commands on any database in your log-on account. To use the COPY or FOrM commands on a database outside your account, it must have been released using DBUTIL. If it hasn't, you must know the database's maintenance word, and have read and lock access to the database (the group and account file security must allow read and lock access to ANY).
- ❑ **If you do not have AM capability**, you can use COPY or FOrM on a database outside your log-on group if you know its maintenance word and have read and lock access to it.
- ❑ Any user can execute the SCHema command. However, password information is displayed only for users who have signed on as the database creator, Account Manager (AM) or System Manager (SM).

Logging

Database logging must be disabled before DBMGR can open a database. Use Hewlett-Packard's DBUTIL program to do this.

Intrinsic Level Recovery

Intrinsic level recovery (ILR) must be disabled before you use the DBMGR DELeTe, CAPAcity, ERAse or RELOad commands. It must be re-enabled after any of these commands is performed. Use Hewlett-Packard's DBUTIL program to do this.

Third Party Indexing

Under TurboIMAGE version C.04.03, TurboIMAGE intrinsics were enhanced to accommodate third party indexing packages, such as OMNIDEX, under Hewlett-Packard's Standard Interface to Third Party Indexing (TPI). TurboIMAGE intrinsics can now perform searches — as well as automatic, real time updates — on the indexes of databases that are enabled for TPI. As beneficial as this feature is from a programming standpoint, it can create problems for data administrators.

- ❑ Sometimes, the real time updates of indexes can cause a performance bottleneck. In a reload, for example, as the intrinsics erase the entries in a detail set, they also erase any associated key values present in indexes. Then, as the set is reloaded, the intrinsics add the same key values back into the indexes as they reload each entry.
- ❑ Other operations on TPI-enabled databases, such as renaming or copying a database, may require you to re-enable databases for TPI.

DBMGR recognizes whether or not a database has been enabled for TPI and prevents both types of problems discussed above. For example, DBMGR bypasses Third Party Indexing on a database before performing RELOads, or CAPAcity changes. Also, when copying or renaming a TPI-enabled database, DBMGR enables the renamed database, or copy of the database, for TPI.

Released databases

Databases that have been released using DBUTIL must be re-released after a CAPacity change, REName, COPy, DELeTe or RELoad operation is performed on them.

DBMGR creates a new file for the resized, copied or reloaded data set. That file is left secured because there is no supported intrinsic for releasing privileged files. (Although DBMGR uses privileged mode, no unpublished or unsupported intrinsic calls are used.)

After you perform a CAPacity change, the database is secured. If you do not release it, and subsequently attempt to copy the database, the only message that appears is an FOPEN error. To avoid this situation, **always re-release the database after performing a DBMGR CAPacity change.**

Break/abort

DBMGR disables [ctrl]-Y during processing. Break/Abort is allowed, but is not recommended. If you perform a Break/Abort during a CAPacity change, ERAse, DELeTe or RELoad operation, you must restore the database from tape.



Always backup a database before using the ERAse, RELoad, DELeTe or RENAmE commands. A backup is strongly recommended before using the CAPacity change command. A system failure or process abort during these operations corrupts the database. If this happens, it must be restored from a backup copy.

Concurrent processes



Do not run DBMGR against two or more databases simultaneously from the same log-on group. DBMGR creates a file in your log-on group to use in performing certain functions. Concurrent processes of DBMGR in the same group may cause conflicts.

Structural problems

To fix a broken chain in a detail set, use the CAPacity command on each master that has a path to the detail and re-specify each master's current capacity. This rehashes the masters' entries, repairing any synonym chain damage that would interfere with a successful reload of the detail set.

Then, use the RELoad command to reload the detail. This fixes any broken chains, linking the detail entries to their master(s) or to each other.

During a REload operation, DBMGR logs any inconsistencies, like a detail entry without a corresponding master entry, by writing the detail entry to an MPE file. DBMGR then adds any master record(s) that are needed to enable the entry to be reloaded.

DBMGR issues a warning message if a REload operation encounters any such inconsistencies. It displays the number of master entries that had to be added and the name of the file that contains the suspect detail entries. To list this file, enter:

```
FILE LP;DEV=LP
FCOPY FROM=DBMGRX00;TO='LP;OCTAL;CHAR
```

The maximum number of entries the file can contain is 10,000. Those entries can subsequently be examined to determine whether they contain corrupted data. If they do, you can locate and correct the corresponding entries in the database. This added level of protection makes DBMGR faster and easier to use than DBUNLOAD/DBLOAD, and safer if database damage is suspected.

IMAGE and TurboIMAGE

To use the DBMGR CAPacity, ERAse or RELoad commands, both the system intrinsic and the database must be the same: both must be IMAGE, or both must be TurboIMAGE. The DElete command is supported for TurboIMAGE use only. You can use all other DBMGR commands regardless of whether the system intrinsic or the database is IMAGE or TurboIMAGE.

If you have a TurboIMAGE database with many data sets, you may not have enough system resources to open the sets during a FOrm, COpy or REName operation. If an Unable to open file error message appears, run DBMGR with the ;NOCB option.

Job mode processing

You can run DBMGR either in session mode or in job mode. Job (or batch) mode is useful for lengthy operations, such as a reload of a large detail set, or a capacity change on a large master.

In job mode, DBMGR operations are performed without dedicated use of a terminal. They can also be scheduled, or initiated from a remote location without the possibility of a phone line drop interrupting an operation.

DBMGR works almost identically in job or session mode, with the following exceptions.

- ❑ In job mode, DBMGR does not ask permission to proceed, as it does for some operations in session mode. For example, when it erases multiple data sets in session mode, DBMGR requests final verification for the erasure. This request is not made in any job mode processing.
- ❑ If an error occurs in job mode, DBMGR immediately sets the JCW (job control word) to fatal, and terminates. Possible errors include improper commands, incorrect responses, or errors resulting from an external condition (for example, an existing file that is blocking a copy operation). This reduces the risk of a job executing improperly due to an unexpected condition.

Sample job streams for the CAPacity, RELoad and ERAse commands are shown next. Note that the comments that are shown enclosed in parentheses are not allowed in an actual job stream. Also note that you may need to insert the maintenance word for the database on a separate line after the database name, depending on your user capabilities and where the database resides. See “DBMGR and security” on page 3-37.

Sample job streams

```

!JOB DBCAP,MGR.ACCOUNT
!RUN DBMGR.PUB.DISC
MYBASE                                     (Database name)
CAP ACCT-MASTER=1023                       ((DBMGR command)
QUIT                                       (DBMGR command)
!EOJ

```

```

!JOB DBREL,MGR.ACCOUNT
!RUN DBMGR.PUB.DISC
MYBASE (Database name)
REL ACCT-DETAIL (DBMGR command)
QUIT (DBMGR command)
!EOJ

!JOB DBERA,MGR.ACCOUNT
!RUN DBMGR.PUB.DISC
MYBASE (Database name)
ERA ACCT-DETAIL (DBMGR command)
QUIT (DBMGR command)
!EOJ

```

INFO string processing

You can run DBMGR with an INFO= string to provide a database name and commands. The string must contain every input that would be required in a job stream. This includes:

- the database name
- a maintenance word (if any)
- a command
- input for the command (if any)
- an EXIT or QUIT command

The INFO= string can contain multiple commands and/or multiple databases, if every required input is present. The length of the string and use of command continuation characters is determined by the operating system.

The parameters (*dbname*, *command*, etc.) must be separated from one another in the INFO= string by a backslash (\) character. Leading and trailing blanks are allowed. For example, the following command runs DBMGR with an INFO= string:

```

RUN DBMGR.PUB.DISC;INFO="SALES.DEMO \ CAP;&
DEV=3 \ ORDER-LINES \ 20000 \ QUIT"

```

This INFO= string opens the SALES database, starts the CAPacity change operation, and specifies that the set to be resized should reside on device number three (DEV=3). Then, it selects the ORDER-LINES data set, sets

the capacity to 20000 and exits the program. Notice that an ampersand (&) is used as a command continuation character in MPE.

An INFO= string is useful for UDCs that run DBMGR. The following example shows a UDC that performs a CAPacity change.

```
CAP !BASE, !SET, !CAPACITY
RUN
DBMGR . PUB . DISC ; INFO = " !BASE\CAP\ !SET\ !CAPACITY\QUIT "
```

If you wanted to change the capacity of the INVENTORY data set in the SALES database to 20,000 records, you could enter:

```
CAP SALES.DATA,ORDER-LINES,20000
```

To set up a UDC that can run DBMGR with or without an INFO= string, copy the following example:

```
DBMGR INFO="NULL"
RUN DBMGR.PUB.DISC;INFO="!INFO"
```

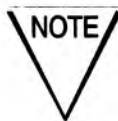
3

DBMGR commands

DBMGR is a command driven program. You can enter any command, or its abbreviation at the DBMGR: command prompt. All commands apply to the currently open database, which is declared when running DBMGR, at the database name prompt, or referenced in a BASE command string. There are two ways to enter commands.

- ❑ The easiest way to use DBMGR is to enter its commands by themselves, with no parameters. When you use this method, DBMGR prompts you for any additional information that may be necessary. The only exception to this is the BASE command, which requires you to append the name of a database. To exit from any other prompt, enter E. To get help, enter H or ?.
- ❑ The quickest way to use DBMGR is to enter the command with accompanying parameters. For example, to increase the capacity of the CUSTOMERS data set by 100 records, you could enter:

```
CAP CUSTOMERS=+100
```



Some DBMGR commands use an OMNIDEX option or ;ODX parameter, which only affect OMNIDEX version 2 indexes.

Listing DBMGR commands

After you open a database, DBMGR issues a `DBMGR:` command prompt. To display a list of DBMGR commands, enter `HELP`, `H` or `?` at the `DBMGR:` command prompt.



You can enter any of the DBMGR commands as abbreviations. Only the `BASE`, `COPY`, `FORM` and `SCHEMA` commands can be executed across MPE accounts.

The DBMGR commands are listed below. You can enter them as their initials (shown in capital letters).

<code>BASE</code>	changes the active database.
<code>CAP[acity]</code>	changes the capacity of a specified data set, all data sets, or a complex of one master and all details with a path to it.
<code>COP[y]</code>	copies the current database.
<code>DEL[ete]</code>	deletes a data set, or orphaned data item.
<code>ERA[se]</code>	erases a data set, a data set's indexes or all OMNIDEX indexes.
<code>E[xit]</code>	exits the <code>DBMGR:</code> command prompt and asks for a new database name.
<code>FO[rm]</code>	gives a form display for all data sets (like a <code>QUERY FORM SETS</code> listing), or displays information about a specified data set.
<code>H[elp]</code>	displays help information, or lists possible responses for individual command prompts, or displays command information at a <code>DBMGR:</code> command prompt.
<code>PRI[mary]</code>	changes a detail data set's primary path.
<code>QUIT</code>	exits DBMGR and returns to the operating system (<code>:</code>) prompt.
<code>REL[oad]</code>	reloads a data set, or all detail data sets in primary path order; also repairs any broken chains as it reloads.
<code>REN[ame]</code>	renames the database, a data set or a data item.
<code>SCH[ema]</code>	generates a schema from the current database.

Each of the commands listed above, and their parameters, is discussed next under its own heading.

The BASE command

BASE *dbname*[.*group*] [*.account*]

The BASE command lets you open a database for processing by DBMGR. If a database is already open, DBMGR closes it before opening the new database. You need not be logged on as the database creator to use the BASE command.

Enter the BASE command followed by a space and the name of the database you want to open. For example, to open a database named SALES in a group named TESTDB, enter BASE SALES.TESTDB.

The BASE command parameters are defined below.

- dbname* is the name of the database you want to open.
- group* is the group where the database resides, if you want to open a database that is outside of your log-on group.
- account* is the account where the database resides, if you want to open a database that is outside of your log-on account.

The CAPacity command

CAP[acity]=[±]{ @/ *setname*}[*n*][%[:|F>*n*%]];DEV=#



NOTE

You must be logged on as the database creator to use the CAPacity command.

The CAPacity command lets you change the capacity of any data set. You can specify an increase or decrease as a number of records or as a percentage. Additionally, you can change set capacity based on entry count. For example, you could increase by 10% the capacities of all data sets that are more than 80% full.

You can use the following parameters with the CAPacity command.

- setname* specifies the data set on which to perform the capacity change.
- @ changes the capacities of all data sets in the database. This parameter is often used in combination with the IF>n% parameter.
- n changes the capacity of a set to a number or entries (n). This option is allowed only when maintaining a single data set.
- ±n increases/decreases the capacity of the set by n entries. This option is allowed only when maintaining a single data set.
- n% sets capacity such that the resulting data set(s) will be n% full. For example, 80% makes the data set(s) 80% full.
- ±n% increases/decreases the capacity of the set(s) by n%.
- ;IF>n% applies the capacity change to any sets that are more than n% full. For example, IF>80% means all data sets that are more than 80% full.
- ;DEV=# moves the data set(s) being resized to a device designated by the logical device number (#).

Below are examples of how to use these parameters.

CAP CUSTOMER-NOTES ;DEV=3
Places the CUSTOMER-NOTES set on disk device 3.



Using the ;DEV=# option lets you place detail sets on a different drive from the master; doing this can improve performance.

CAP CUSTOMER-NOTES=40000
Changes CUSTOMER-NOTES' capacity to 40,000 records.

CAP CUSTOMER-NOTES=-400
Reduces CUSTOMER-NOTES' capacity by 400 records.

`CAP CUSTOMER-NOTES=80%`

Sets CUSTOMER-NOTES' capacity such that the set is 80% full.

`CAP CUSTOMER-NOTES=+10%`

Increases CUSTOMER-NOTES' capacity by 10%.

`CAP CUSTOMER-NOTES=+10%;IF>80%`

Increases CUSTOMER-NOTES' capacity by 10% only if the set is more than 80% full.

`CAP @+=20%;IF>70%`

Increases any data set's capacities by 20% only if it is more than 70% full.

You can also enter the CAPacity command without parameters. If you enter the CAPacity command with no parameters, you are prompted for them. First, DBMGR asks you what you want to resize. Enter H or ? to list all the sets in the database, enter the name of a data set, or enter @ to resize all data sets. After you specify a data set name, DBMGR prompts you for the new capacity.

The minimum capacity for a master set is the current number of entries in the set, unless the set is empty, in which case the minimum capacity is one (1). However, the minimum capacity recommended for a hashed master should be 20% greater than the projected number of entries. When you change capacity to some entry count, DBMGR automatically sets capacity at the next highest prime number.

The minimum capacity for a detail data set is equal to the highest number of entries that have occupied the data set since the last reload, or since the data set was created (the "high water mark"). After reloading detail sets, then, you may want to change their set capacities as well.

The maximum number of entries that TurboIMAGE allows per set is 2,139,094,785.

After you perform a capacity change on a previously released database, it is secured. If you do not release it, and attempt to copy it, an FOPEN error is returned. To avoid this situation, always re-release the database after you perform a CAPacity change.

The COPY command

COPY[y]

The COPY command lets you copy the structure and data of a database. During this process, you can assign a new name to the copy. DBMGR copies about 30,000 sectors per minute.

You need not be the database creator to use the COPY command. DBMGR prompts you for a maintenance word if you are not the database creator or system manager. See “DBMGR and security”, on page 3-37 to determine which databases you can access from an account.

To copy a database, enter **COPY** at the **DBMGR:** command prompt. When the **Name to be assigned to the new copy?** prompt reappears, you can enter a different name, press **[return]** to keep the same name, or enter **//** to return to the **DBMGR:** command prompt. DBMGR checks for a conflict between the name you give the copy, and any existing files. If a conflict exists, DBMGR lists all of the files that are blocking the COPY operation, and prompts again for a name for the copy. The new database can be copied to a group other than your log-on group, provided that you have MPE write access to the group.

For example, if your original database was **SALES.DEMODB.DISC**, you could copy it to the **TEST** group of your log-on account like this:

```
Name to be assigned to the new copy? SALES.TESTDB
```

DBMGR does not change the active database after a copy operation. To change databases, enter the **BASE** command, as described on page 3-45.

The DELeTe command

DEL[ete][SET *setname*/ ITEM *itemname*]



The DELeTe command is not supported on IMAGE/3000 databases. It is supported only on TurboIMAGE databases.

The DELeTe command removes data sets, unreferenced (orphaned) data items, and indexes from a TurboIMAGE database. The DELeTe SET command removes a data set from the database. The DELeTe ITEM command removes an orphaned data item from the database.

You can use the following parameters with the DELeTe command:

SET *setname* deletes the data set referenced in *setname*.

ITEM *itemname* deletes the data item referenced in *itemname*.

Below are examples of how you can use these parameters.

DELETE SET ORDER-LINES
Deletes the ORDER-LINES set.



DBMGR will not delete a master set with paths to detail sets. When deleting the last set in a database, DBMGR asks if you want to delete the TurboIMAGE root file as well.

DELETE ITEM ADDRESS2
Deletes the ADDRESS2 item, if it does not occur in any data sets.

If you enter the DELeTe command with no parameters, DBMGR asks what you want to delete.



To remove the contents of a set, use the ERAse command. **To remove a set (or item) itself**, use the DELeTe command. **To remove the OMNIDEX indexes from a database**, use OmniUtil's 5. Remove All Indexes option from the Index Installation menu.

The ERAse command

ERA[se] [*setname*][:DOMAIN]



You must be logged on as the database creator to use the ERAse command.

The ERAse command lets you erase (remove all entries from) a manual master or detail data set. You cannot explicitly erase Automatic masters. As in TurboIMAGE, you can only erase a master set when all the detail sets with paths to it have been erased. The ERAse;DOMAIN command erases the master and all the details with a TurboIMAGE path to it. This command is described below. You can erase any detail data set, regardless of the number of master data sets with paths to it.

You can use the following parameters with the ERAse command:

setname represents the data set to be erased.

;DOMAIN instructs DBMGR to erase the master data set specified by *setname*, as well as all of the detail data sets with a path to it.

Below are examples of the ERAse command and its parameters.

ERASE CUSTOMERS

Erases the CUSTOMERS data set.

ERA CUSTOMERS;DOMAIN

Erases all of the detail data sets with a path to CUSTOMERS, then erases the CUSTOMERS data set.

If you enter the ERAse command with no parameters, or the ;DOMAIN option alone, DBMGR prompts you for a data set's name. Enter H or ? to list all the sets in the database. Enter E to exit the command.

Depending upon the capacity of the set, a DBMGR ERAse operation is over 50 times faster than performing a QUERY FIND ALL;DELETE operation.

The Exit command

E[xit]

Use the Exit command to close the active database and exit DBMGR. Simply enter E at the DBMGR: command prompt, and the database name prompt appears. To return to DBMGR, enter the name of a database. To exit DBMGR, press **[return]** at the database name prompt. You need not be logged on as the database creator to use the Exit command.

The FOrM command

The FOrM command generates a list of the data sets in the active database. The list includes vital statistics such as capacity, number of entries and percent of capacity. Anyone can use the FOrM command.

A DBMGR FOrM listing is like a QUERY FORM SETS listing. However, the FOrM command also displays a percent of capacity for each data set.

You can use the following parameters with the FOrM command:

- @ instructs DBMGR to display all data sets. @ is the default, so you can omit it if no other options are used.
- setname* specifies the data set to be displayed if a single set is requested. The set can be specified across groups and accounts like, *setname.group.account*.
- IF>*n*% instructs DBMGR to display data sets only if they are greater than *n*% full.

Below are examples of how you can use these parameters.

FORM CUSTOMERS

Displays information about the CUSTOMERS data set.

FORM;IF>90% or FORM @;IF>90%

Displays a form listing of all data sets that are more than 90% full. The form listing might look like this:

```
Data base: SALES.DEMODB.OSBDEV                    FRI, OCT 7, 1994, 11:11 AM
```

Sets:	Type	Item Count	Capacity	Entry Count	Entry Length	Blocking Factor	%Full
CUSTOMER-NOTES	D	5	3360	3336	38	24	99.2
INVENTORY	D	5	200	184	21	40	92.0

The Help command

H[elp] [*command*]

Use the Help command to get help at any prompt. Any user can use the Help command.

If you enter H at the DBMGR: command prompt, a list of DBMGR commands appears. If you enter H at a subprompt for a command, a message about the information that DBMGR is expecting appears. When you enter H followed by the name of a command, DBMGR displays help about a specified command.

The PRImary command

PRI[mary]

The PRImary command lets you change the primary path of a detail set. You must be logged on as the database creator to use the PRImary command.

The PRImary command is entered with no parameters. When you enter PRI at the DBMGR: command prompt, DBMGR asks you for the name of the data set whose primary path you want to change. After you supply the name of a data set, DBMGR asks you for the new primary path. This must be a TurboIMAGE search item, which constitutes a path to some master.

A sample PRImary operation is shown below:

```
DBMGR: PRI
Change a data set's primary path
Data set name? ORDER-LINES
Primary path: PRODUCT-NO
New primary path: CUSTOMER-NO
```

Note that although the primary path is changed, the locations of the records in the detail set are not. To reorganize the records in sorted order by the new primary path, perform a REload operation.

The QUIT command

QUIT

Use the QUIT command to exit from DBMGR and return to the operating system (:) prompt, or to the calling program.

The REload command

REL[oad] [[:DEV=#][:SORT=*field1,field2,...*]]



The ;SORT and ;DEV options are allowed only when specifying a data set name, not when specifying @.

The REload command lets you reload the entries in a detail set in primary path sequence. You should perform a REload operation after a PRImary operation. REload also repairs any broken chains and repacks the data set, removing any empty slots that were left from deleted entries. You must be logged on as the database creator to use the REload command.

You can enter the REload command using any of the following parameters.

@	instructs DBMGR to reload all detail data sets in the currently opened database.
<i>setname</i>	specifies the data set to be reloaded in primary path order.
;DEV=#	instructs DBMGR to place the new, reloaded detail data set on a disk drive with the logical device number #. This option is allowed only when specifying a data set name.
;SORT= <i>field1, field2...</i>	instructs DBMGR to apply an alternate or secondary sort using the fields specified by <i>field1, field2</i> , etc. This option is allowed only when specifying a data set name, not when specifying @.

The following are examples of how you can use these parameters.

RELOAD @

Reloads all detail data sets in the currently opened database along their primary paths.

REL ORDER-LINES;DEV=3;SORT=DELIVERY-DATE

Reloads the ORDER-LINES detail data set in primary path order, and uses DELIVERY-DATE as a sort field. The set is reloaded onto the disk drive that was assigned the logical device number 3. For more information about REload sorting, see "Data sorting", below.

If you enter the REload command with no parameters, you are prompted for the name of the set to reload. Note that you can append the ;DEV= and ;SORT= options to the name of a data set.

If you are using REload to fix a broken chain, you should first use the CAPacity command to change the capacity for each master that links to the detail. This rehashes the entries in the masters, repairing any synonym chain damage that would interfere with a successful reload of the detail set.

Data sorting

The REload command sorts detail data set records in primary path sequence. It also sorts by the TurboIMAGE sort field, if one exists within the data set.

RELOAD does not maintain entries in the chronological order in which they were entered unless the primary path includes a sort field that maintains the order. If your application requires that data be kept in chronological sequence, assign a sort path through TurboIMAGE, or periodically REload the set and use the ;SORT option to sort the records by a time or date field.

If the data set that is being reloaded has no TurboIMAGE sort field, you can specify fields for sorting within primary path sequence. For example, if the primary path for an ORDERS data set is ACCOUNT-NO, and the data should be in sequence by DATE within ACCOUNT-NO and by LINE-ITEM within DATE, you would enter the following REload command:

```
REL ORDERS;SORT=DATE,LINE-ITEM
```


You can specify up to seven temporary sort fields. These fields are used to sort entries only during the reload operation, and are not maintained as TurboIMAGE sort fields as new records are added. When a TurboIMAGE sort chain is present, it takes precedence over any sort fields you specify through the ;SORT option.

Set capacity

After a REload operation has been performed, the capacity of a detail set can be reduced to its current entry count because the TurboIMAGE end of file is the current entry count. Thus, a REload operation is useful after archiving or deleting detail records, to allow a reduced set capacity to reduce the amount of disk space used. For further details, see “The CAPacity command”, on page 3-45.

The REName command

REN[ame]

The REName command lets you rename the database, or any data sets or data items. When you rename a database, its index files are automatically renamed as well. If the database was enabled for TPI, the renamed database is also enabled for TPI. You must be logged on as the database creator to use the REName command.

When you enter the REName command, DBMGR first prompts you to enter **B**, **S** or **I** to indicate whether to rename a **dataBase**, **Set** or **Item**:

```
Rename the database, a data set, or a data item
(Base/Set/Item/[Exit]):
```

When renaming a database, you can move it into any group to which you have file write and save access by adding that group’s extension to the new name. For example, to move the SALES.PUB database into the TESTDB group of your log-on account, rename it “SALES.TESTDB”.

DBMGR checks for conflicts between any existing files and the files that are required by the renamed database. If a conflict exists, DBMGR lists all of the files that are blocking the REName operation, and prompts again for a new name. When you are prompted again for a new name, you can do either of two things.

- Enter a different name.
- Press **[return]** and exit DBMGR. Rename or purge the conflicting files. Then run DBMGR and REName the database.

When you are using the RENAME command to rename data sets, DBMGR prompts for one set at a time. When you enter a data set name, DBMGR prompts for its new name. If you press [return] without entering a new name, the set name is left unchanged.



You can enter H@ at the Data set name: prompt to list all the sets in the database. You can enter // or E at any time to exit the command and return to the main DBMGR: command prompt. You can also enter H or ? to display help.

The RENAME operation works the same for items, except that you can get a list of items that begin with a specified string by entering H, one blank, and the string. For example, H A would list all items that begin with the letter, "A", and H ACCT would list all items that begin with the string, "ACCT".

The SCHEMA command

SCH[ema][:NOPASS]

The SCHEMA command lets you decompile the current database's TurboIMAGE root file and generate a schema from it.

You can use the following parameters with the SCHEMA command:

- ;NOPASS** instructs DBMGR to suppress all security passwords and read/write lists that would normally appear on the screen, printer or file when the SCHEMA command is issued.
- LP** instructs DBMGR to send the SCHEMA command's output to the line printer.
- filename** instructs DBMGR to send the SCHEMA command's output to a named MPE file, *filename*. You can specify a group with this file name (*filename.group*) if you want the file to reside in a group outside of your log-on.
- TERMINAL** instructs DBMGR to send the SCHEMA command's output to the terminal screen.

The following are examples of how you can use these parameters.

SCH;NOPASS LP

Creates a schema, suppresses all security passwords and read/write lists that would normally appear, and routes the resulting text to the system line printer.

SCH SALESSC.SOURCE

Creates a schema and sends the output to an MPE file named SALESSC in the SOURCE group.

If you enter the SCHEMA command with no parameters, you are prompted for a destination. You can list the schema to the terminal by pressing **[return]**, to the line printer by entering LP, or to a file by entering the name of a file. If you specify a file that already exists, you are asked whether you want to purge the file's existing contents.

You need not be the database creator to use the SCHEMA command. However, password information is displayed only if you are the database creator, or if you have SM or AM capabilities.



Contributed OMNIDEX Utilities

This section describes the contributed utilities included with OMNIDEX. These utilities, which reside in the UTIL group of the DISC account, are not supported by DISC's Technical Services staff, but are useful for a variety of data management tasks. They include the following.

- | | |
|---------------------|--|
| ALTPROG | is a command file used by XLSWITCH. |
| CAP ² | increases index file capacities for a specified database. |
| CAPCHK ² | displays the index file capacities for a specified database. |
| COBGEN | generates COBOL source code. DISC's version of this popular utility generates additional element definitions (like a 21-word <i>status</i> array) for use with ImagePlus intrinsics. |
| FUTIL | is used to copy, modify (change maxdata or capabilities), purge, or rename files, especially program files. When using the Modify command, you can target files individually, or collectively by using a wildcard (@) operator when specifying a group or account. |
| IDADD | supplies OMNIDEX ID values for records in a sets to which a J2 SI has been added. IDADD is discussed in detail in the "Streamlining OMNIDEX Databases" section of the "Topics" chapter. |
| ODXM210 | add this XL to XL.PUB.DISC210 to make it work with XLSWITCH. |
| ODXVER | reports the OMNIDEX version of the XLOMNIDX intrinsic library you are currently running. |

PCLINK	is used for data communication across PCs by DISC Response Center representatives.
PSCREEN	lists terminal memory contents to the line printer.
REGTEST	checks OMNIDEX license information.
SHOWDISC	displays the accounts that take up the most disk space, as well as how much disk space is available.
SQUISHER	compresses and expands files to optimize storage.
SOS	copies segments from one SL to another SL.
TPIDRVR	calls DBFIND and DBGET through a command driven interface. Works well for testing the TurboIMAGE retrieval interface or Standard TurboIMAGE/iX Interface to Third Party indexing applications.
TPIXLCFG ²	alters the XL path of DISC utilities for either TPI-enabled databases, or non-TPI databases.
XLSWITCH	replaces XL.PUB.DISC in situations where there are more than one copy of XLOMNIDX. It checks the <i>dbnameOX</i> file of a database to see which copy of XLOMNIDX to use when accessing the database.
XM	tells whether or not a file is attached to the transaction manager.

-
2. These utilities or command files must be edited to run on MPE/iX versions prior to 5.0. They must contain the DISC account name. Change the first occurrence of XXXXXXXX to the DISCvrr (the name of the current DISC account).

Chapter 4: Topics

This chapter contains general information about OMNIDEX. It contains the following topics:

- ❑ “OMNIDEX Retrievals”, on page 4-2, discusses the many search operations supported for the OMNIDEX Intrinsic Interface and the Standard Interface to Third Party Indexing.
- ❑ “Determining Retrieval Requirements”, on page 4-28, discusses what to look for in your TurboIMAGE database to help determine how to key it for keyword and sorted access.
- ❑ “Updating OMNIDEX Data”, on page 4-50, discusses how to update OMNIDEX databases and their corresponding index structures.
- ❑ “Linking Details to Masters”, on page 4-56, discusses the effects of linking details to a master set during OMNIDEX installation.
- ❑ “Multifind”, on page 4-63, discusses a feature of keyword keys that supports relational searches across unlinked sets and databases.
- ❑ “Maintenance”, on page 4-68, describes maintenance operations that maximize OMNIDEX’s performance. It also discusses situations that can upset OMNIDEX, and how to recover from them.
- ❑ “Other Performance Considerations”, on page 4-97, discusses things you can do to improve OMNIDEX performance.
- ❑ “Streamlining OMNIDEX Databases”, on page 4-103, discusses how you can use OMNIDEX keys to create a more efficient database design.

Please note that most of the examples in this section refer to the SALES.DEMODB.DISC database. To experiment with this database, enter @SALES in upper case letters when you are asked for a password.

OMNIDEX Retrievals

This section discusses the retrieval capabilities that OMNIDEX provides. These retrieval capabilities comprise two different types of access:

- ❑ keyword access
- ❑ sorted access

If you want to try some of the access capabilities discussed in this section, run *DataView*, as discussed on page 3-21. Many of the examples in this section explain how to perform a sample search using *DataView*.

Please note that there are two interfaces that you can use to develop OMNIDEX retrieval applications:

- ❑ the OMNIDEX Intrinsic Interface (see page 4-20)
- ❑ the Standard Interface to Third Party Indexing (see page 4-24)

Each of these interfaces, their retrieval capabilities, and their advantages are discussed in this section. For information about installing these interfaces, see Appendix C.

Keyword searches

The data indexed for keyword keys consists of individual keywords and the identifiers of the records that contain them. When you reference a key, like *CUSTOMER-NAME*, and pass a keyword argument, like *Information*, to a search intrinsic, like *ODXFIND* or *DBFIND*, it locates all the records that have that value indexed for the referenced key. It doesn't matter where the word occurs in the keyed field.

For example, a keyword search on a *CUSTOMER-NAME* keyword key, using the argument *Information*, would find any of the following *CUSTOMERS* records:

```
Information Xpress  
Dynamic Information Systems  
Ready Reference and Information
```


You can search for records using several keyword arguments combined in various operations. This string of keyword arguments and operations used in a search of a keyword key is called a *keyword list*. It is passed through the *keywords* parameter of an ODXFIND call, or the *argument* parameter of a DBFIND call. The arguments and operations supported for keyword searches are discussed later.

Qualifying records

When ODXFIND or DBFIND qualifies records, it returns a count of the number that qualified. This count is called a *qualifying count*, and corresponds to the `Qualified records:` message at the bottom of a DataView record selection window. These stored record identifiers are collectively called an *internal ID list*.

When a keyword argument does not exist for the referenced key, and the operation it is used in does not qualify any records, an error is returned, `No entries contain <keyword>`.

If a keyword list does not qualify any records even though all keywords exist, the TurboIMAGE and OMNIDEX condition words are set to zero to indicate a successful call, and the qualifying counts in the *status* array are set to zero. This is consistent with a traditional mode 1 TurboIMAGE DBFIND, which succeeds if the specified search item exists, even if the chain count is zero.

Referencing the internal ID list to progressively qualify records is discussed next.

Searching across keys

OMNIDEX lets you progressively qualify records using different arguments and operations against one or more keys until you have the exact subset of records you want. When progressively qualifying records by searching several keyword keys, you must reference the internal ID list at the time of the keyword search. You can reference the internal ID list, and continue to refine that set of records, even after you've retrieved or processed the records. The internal ID list remains in memory until you overwrite it with a new search, or you call DBFIND mode 13 to back out to the previous internal ID list.

You can reference the most recent internal ID list in calls to DBFIND or ODXFIND mode 3 or 5 by beginning the keyword list with an AND or OR operator, followed by additional keywords. You cannot enter a Boolean operator alone without incurring a Missing keyword in the list error.

The leading operator that you use determines what effect the current keyword search has upon the internal ID list. A leading AND intersects the internal ID list with the record IDs qualified by the current keyword search. A leading OR unites the internal ID list with the record IDs qualified by the current keyword search. A leading AND NOT excludes the record IDs qualified by the current keyword search from the internal ID list.

Another way to reference the most recent internal ID list is to use an asterisk (*) at the beginning of the keyword list. If you enter this operator alone, it simply references the last internal ID list and maintains the last qualifying count. This is what DataView does when you press **[return]** to skip a search prompt, or enter * at a search prompt after changing tables. You can also combine the asterisk with additional keyword arguments in a Boolean operation. For example, * NOT DATA entered for a CUSTOMER-NAME key references the last list of records and exclude those with "DATA" in the CUSTOMER-NAME field. Note, however, that the asterisk must always begin the keyword list to operate on the most recent internal ID list.



A leading NOT operator in a keyword list begins a new search that overwrites the internal list, if one exists.

If there is no asterisk (*), or leading AND or OR operator, in a keyword list, OMNIDEX assumes that a new search is underway and overwrites the existing internal ID list.

A good example of progressively qualifying records would be looking for all the CUSTOMERS in California who purchased from sales person HWS since January of 1992. Suppose that the STATE, SALESMAN, and LAST-PUR-DATE keys have not been grouped. Therefore, a separate intrinsic call is required to search each one.

The progress of calls, and the contents of certain parameters, would look like this. Note that the *status* words return the qualifying count:

DBFIND mode 12	<i>Dset</i>	<i>Item</i>	<i>Argument</i>	<i>Status</i> words 5-6
ODXFIND mode 5	<i>Dset</i>	<i>Field</i>	<i>Keywords</i>	<i>Status</i> words 12-13
First call	CUSTOMERS;	STATE;	CA;	283
Second call	CUSTOMERS;	SALESMAN;	AND HWS;	281
Third call	CUSTOMERS;	LAST-PUR-DATE;	AND >9201	14

Table 4-1: Qualifying records using OMNIDEX

Searches across tables

You can search keyword keys that reside in different linked tables (data sets). When sets are linked to each other or to the same master set during key installation, they belong to the same OMNIDEX domain. SI domains and record complexes are discussed on page 4-57 and on page 4-58 respectively.

When progressively qualifying records across tables (sets), you must reference the internal ID list of previously qualified records, as described above. You can then use other keyword arguments and operations to progressively qualify records based on the data in the related sets.

When you search across tables, you must consider whether record complex or record specific keys are involved. This is discussed in "Split Retrievals", on page 4-60.

Null selections

When you search across keys or across tables and ODXFIND or DBFIND return a qualifying count of zero, they automatically maintain the most recent list of qualified records. Consequently, online applications can reprompt the user when the search criteria that they enter yields a null selection.

In batch mode, where re-prompting is not possible, a null selection generally should not be backed out to the most recent list of qualified IDs. To disable the back-out feature for the OMNIDEX Intrinsic Interface and drive the internal list to zero on a null selection, call ODXFIND with mode option 400 or 700. To disable the back-out feature for the Standard Interface to Third Party Indexing, call DBCONTROL in mode 801 before calling DBFIND.

Retrieving records

When you have qualified the records you need, you can retrieve the records. How an application does this depends on whether it uses the OMNIDEX Intrinsic Interface or the Standard Interface to Third Party Indexing.

In the OMNIDEX Intrinsic Interface following a call to ODXFIND, retrieve search items (SIs) from memory by calling ODXGET. ODXGET returns these SIs, or relative record numbers, to the calling application. The application then calls DBGET, which uses the SIs or relative record numbers to retrieve the qualified records for further processing.

In the Standard Interface to Third Party Indexing, after a call to DBFIND, simply call DBGET to retrieve the qualified records. DBGET mode 5 reads forward through the list of qualified records. DBGET mode 6 reads backward through the list of qualified records.

Keyword arguments

As mentioned earlier, keyword arguments are passed through the *keywords* parameter of ODXFIND, or the *argument* parameter of DBFIND. The types of arguments supported for keyword searches are discussed below.

Soundex Phonetic Searches

You can perform phonetic searches on keyword keys installed with the Soundex option. To specify a phonetic, or Soundex, search on a designated Soundex field, append the Soundex operator (!) to the keyword(s) in question. For example, to find all records with the keywords "ALLEN", "ALAN" or "ALAINE" indexed for a given key, search that key using the argument ALAN!.

If a Soundex argument is used on a key that is not installed with the Soundex option, error message -224 Not a Soundex Field is returned. When a Soundex key is grouped with other keys, be sure to reference the Soundex key in the *field* parameter of the ODXFIND call.

For information about installing Soundex, see "Soundex", on page 2-13.

Pattern-matched and generic arguments



Generic and pattern-matched values are not supported for searches on binary keys.

You can use the following wildcard characters anywhere in an ASCII keyword argument value:

- ? represents any single printable character.
- # represents any single digit (0-9).
- @ represents any number of ASCII characters, including spaces.

These wildcard characters let you use incomplete text strings to find records. Here are some examples of pattern matched arguments and the keywords they imply.

ADEL??DE	finds records that contain any eight byte keyword value beginning with "ADEL" and ending with "DE". For example: ADELAIDE, ADELERDE, ADEL27DE.
???LAIDE	finds records that contain any eight byte keyword value ending with "LAIDE." For example: ADELAIDE, KOOLAIDE, 659LAIDE.
??ELA???	finds records that contain any eight byte keyword value with "E," "L," and "A" as the third, fourth, and fifth characters. For example: ADELAIDE, PRELATES, or 72ELAN3F.

???????	finds records that contain any eight byte keyword value.
143##	finds records that contain any five digit number beginning with "143." For example: 14304, 14367, 14393.
###28	finds records that contain any five digit number ending with "28". For example: 10028, 35628, 97228.
1###6	finds records that contain any five digit number beginning with 1 and ending with 6. For example: 13086, 14576, 17146.
#####	finds records that contain any five digit number.

Generic (partially specified) keyword arguments use an at-sign (@) to represent a string of any length contained somewhere in a keyword. You can place the at-sign at one or more locations in a keyword argument to specify a generic string. Below are some examples of generic arguments and the keywords they imply.

MANAG@	finds records that contain any keyword value beginning with "MANAG". For example: MANAGER, MANAGING, MANAGEMENT.
@TEST	finds records that contain any keyword value ending with "TEST". For example: TEST, ATTEST, PROTEST.
@TEST@	finds records with any keyword value that contains the string "TEST". For example: ATTEST, CONTESTANT, TESTING.
TE?T@	finds records with any keyword value beginning with "TE", followed by any printable character, followed by "T", and ending with any string. For example: TERTIARY, TESTING, TEXT, TEXTILE.

As the last example demonstrates, different wildcard characters can be combined in the same keyword argument.

In ranges and relational expressions, you can only place wildcard characters at the end of an argument value. For example, >8904#?<900@ is supported, while >89#?01<90@31 is not supported. Range and relational retrievals are discussed on page 4-12.



Searches where arguments contain leading wildcards are not as efficient as searches where arguments contain trailing wildcards.

Arguments that contain special characters

Enclose search values that contain spaces or special characters in double quotes ("). If a generic keyword or keyword range includes a blank or special character, place wildcards and search operators (such as Boolean, range, and relational tokens) outside the quotes.

For example:

- ❑ " ABC"@ finds all of the keywords that start with a blank followed by ABC.
- ❑ " BC": " DA" finds the range of keywords from those that start with one blank and the letters BC through those that start with a blank and the letters DA.
- ❑ " 1###: " 5### finds the range of four-digit ASCII numeric values that starts with a blank followed by a 1 through a blank followed by a 5.

You should also enclose arguments that contain reserved characters in double quotes to prevent such characters from being parsed as operators. For example, you might search a No Parse PART-NO key using the argument "2392A-092" to find a particular part. Putting quotes around the argument prevents the search intrinsic from interpreting the hyphen (-) as a Boolean NOT token. For information about reserved characters, see "Reserved characters and strings", on page B-6.

You can also use double quotes around argument strings that could be interpreted as literal operators. For example, you might search a STATE key using the argument WA OR "OR" to find records for the states of Washington and Oregon. The quotes around OR, the abbreviation for Oregon, distinguish it from the OR Boolean operator.

To index special characters for a keyword key, install it with the No Parse (NP) option, as discussed on page 2-10.

Converting date arguments

Because dates can be stored in a variety of formats, DBFIND, and ODXFIND modes 3 and 5, include a function to convert between an argument format and a storage format. This function, %DATE, is passed as an expression in the keyword list. This expression can be used like, and combined with, any other keyword in any operation. %DATE expressions use the syntax:

%DATE (*argument-value*, *argument-format*, *storage-format*)

The parameters are:

argument-value represents the date argument value. For example, 900509 or 050990.

argument-format represents the order of the argument-value. The acceptable values are YYMMDD (the default), YYYYMMDD, YY, YYYY, YYMM, YYYYMM, MMDDYY, MMDDYYYY, MMY, MMYYYY, DMMYY and DMMYYYY.

If no value is entered, the entry format defaults to YYMMDD. parameter is positional and must be delimited by commas, as in the examples that follow.

storage-format represents the storage format of the date field on which you are searching. %DATE supports the following formats: YYMMDD, YYYYMMDD, MMDDYY, MMDDYYYY, DMMYY, DMMYYYY, as well as these proprietary formats: PHDATE (PowerHouse internal date format), JDATE (PowerHouse Julian and HP calendar date format) and ASKDATE (ASK Computer Systems internal date format).



When ASKDATE, PHDATE, and JDATE keyword keys are of data type I or J, you must typecast them as type K to support the %DATE function. See "Type Casting", on page 2-11, for more information.

When using a partial date *argument-format* (such as YY, YYYY, YYMM, YYYYMM, MMY, or MMYYYY), the *storage-format* must be ASKDATE, PHDATE, JDATE, YYMMDD, or YYYYMMDD. Also note that partial date entry formats are not supported for range operations.

DBFIND and ODXFIND convert partial date values into an equivalent range operation. So, conversion of partial dates to any other format besides ASKDATE, PHDATE, JDATE, YYMMDD, YYYYMMDD (like MMDDYY) is not currently supported. For example, converting the YYYYMM value "199103" to MMDDYY storage format could result in an ambiguous argument like "03 91", which is useless in a range operation.

Here are some examples of the %DATE function's uses.

- ❑ To convert a YYMMDD date argument into an ASKDATE proprietary date format, use the expression:
`%DATE(920704,YYMMDD,ASKDATE)`
- ❑ You need not specify an entry format when it is the default (YYMMDD). Use double commas to pass a null value for the entry format. So the expression above can be simplified as:
`%DATE(920704,,ASKDATE)`
- ❑ To convert a century date entry format into the typical ASCII YYMMDD storage format, use the expression:
`%DATE(07041992,MMDDYYYY,YYMMDD)`
- ❑ As discussed above, the %DATE function also supports partial entry formats. To convert a partial century date (MMYYYY) into a keyword range to a search a PowerHouse internal date key, use the expression:
`%DATE(071992,MMYYYY,PHDATE)`
- ❑ Each of the %DATE expressions in the examples above can be combined with other keywords in any operation supported for keyword keys. In the example below, %DATE expressions are combined in a range operation:
`%DATE(09301991,MMDDYYYY,YYMMDD) TO`
`%DATE(01011992,MMDDYYYY,YYMMDD)`

Ranges are discussed in the next section.

Keyword search operations

Keyword keys support the following search operations:

- ❑ ranges
- ❑ Boolean operations
- ❑ relational operations

You can combine operations in a keyword list. Ranges and relational operations can be used in Boolean expressions, for example, `>79@ AND <=95@ NOT 85@:89@`. You can't, however, have a relational and a range operation that use the same argument. For example, the argument `>70@:89@` returns an error message: Illegal range operator combination.

The three types of search operations that keyword keys support are discussed next.

Ranges

To search using a keyword range, enter a starting value, a range operator and a stopping value. The default range operator is the colon (:). In DBFIND, and ODXFIND modes 3 and 5, you can use the operators THRU, and TO in place of the colon.

Either the starting value or the stopping value is optional. The syntax is `[startvalue] TO [stopvalue]`, where:

`startvalue TO stopvalue` means all keywords from *startvalue* through *stopvalue*.

`startvalue TO` means all keywords greater than or equal to *startvalue*.

`TO stopvalue` means all keywords less than or equal to *stopvalue*.

For numeric ranges on keys of TurboIMAGE types U, X and Z, the *startvalue* and *stopvalue* both must contain the same number of digits. A range entered as `130:3000` would return ODXFIND error -221 Start and stop values must have the same number of digits.

The range could, however, be entered as `130:999+1000:3000`. A numeric ordering is used to determine the list.

For alphanumeric ranges on keys of TurboIMAGE types U or X, the *startvalue* and *stopvalue* 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.

You can use generic or pattern-matched arguments in ranges on ASCII keys by appending wildcards (@, #, and ?) to a partial *startvalue* or *stopvalue*.

Boolean operations

Keyword keys support combinations of search values in Boolean operations. The operators and their operations are listed in the table below:

Default operator (ODXFIND mode 1)	DBFIND, and ODXFIND mode 3 or 5 operators	Boolean operation
, (comma)	AND or a space	intersects the set of records that contain the keywords it precedes. A key must contain all keywords combined in an AND operation to qualify records.
+ (plus sign)	OR	unites the set of records that contain the keywords it precedes. A key can contain either keyword combined in an OR operation to qualify records.
- (minus sign)	NOT	excludes the records that contain the keyword it precedes. Keys must not contain the keyword preceded by the NOT operator to qualify records. If NOT begins the keyword list in a DBFIND or mode 3 or 5 ODXFIND call, it qualifies all records that do not satisfy the keyword arguments that follow it.

Table 4-2: OMNIDEX Boolean operators

The argument `AC` only qualifies records that begin with `"AC"` (like `"ACE"` and `"ACME"`). `"AARDVARK"`, and `"AZTEC"` would not qualify, because their first two characters do not match the argument value, `AC`. The more characters that you specify, the more precise the retrieval criteria, and the more selective the retrieval.

For the relational operations `"equal to"`, `"greater than"`, and `"greater than or equal to"`, the first value returned is the first matching key value in ascending sequence, or the lowest key value that qualifies. For the relational operations `"less than"`, and `"less than or equal to"`, the first value returned is the first key value in descending key sequence, or the highest key value that qualifies.

For example, an index for a key might contain several key values starting with `"L"`, ranging from `"LABEL"` to `"LUCKY"`. A partial-key `"equal to"` or `"greater than or equal to"` operation on the argument value `"L"` would return `"LABEL"`, while a `"less than or equal to"` operation would return `"LUCKY"`.

Similarly, a `"greater than"` operation using the argument value `"L"` would return the lowest key starting with `"M"`, and a `"less than"` operation using the argument value `"L"` would return the highest key starting with `"K"`. Here are some possible returns of relational retrievals using the argument `L`.

Relational operation and argument value	First key value returned
<code><L</code>	KUDOS
<code><=L</code>	LUCKY
<code>=L</code>	LABEL
<code>>=L</code>	LABEL
<code>>L</code>	MAN

Table 4-5: Sample relational retrievals

Ranges

The Standard Interface to Third Party Indexing supports ranges through modes 1 and 11 of the DBFIND intrinsic. For ASCII sorted keys, you can use TO, THRU, or : as a range operator in a mode 1 DBFIND. You can also combine two relational operations (as in the argument >14<=20). For binary sorted keys, range operations are implied by calling DBFIND mode 11 and defining the *argument* parameter to be large enough to hold two full key values. The first half of *argument* contains the *startvalue*, and the second half contains the *stopvalue*. Remember, you must use full key arguments against binary keys.

The OMNIDEX DBIFIND and DBIGET intrinsics do not directly support ranges. To retrieve ranges of records using the OMNIDEX Intrinsic Interface, find the record that contains the *startvalue*. Then, read sequentially, and programmatically check for the *stopvalue*. The actual operation would involve a “greater than or equal to” or “less than or equal to” using the *startvalue* as the argument, as described earlier. You would then use the *stopvalue* in a conditional statement that tells the retrieval application when to stop retrieving records.

4

Sorted sequential reads

After finding a starting record, by using a relational mode and an argument, you can read forward or backward through the sorted list of qualified records, or re-read the current record, using the DBIGET or DBGET (IMSAM) sorted sequential modes.

The sorted sequential modes are:

DBGET mode (TurboIMAGE)	DBIGET mode (OMNIDEX)	Action
5	90	reads forward through the list
6	91	reads backward through the list
1	92	re-reads the current record

Table 4-6: Sorted sequential reads

See the “DBGET” and “DBIGET” sections in the “Intrinsics” chapter of the *OMNIDEX ImagePlus SDK API Guide* for more information.

The OMNIDEX Intrinsic Interface

OMNIDEX has its own intrinsics, which reside in XLOMNIDX.PUB.SYS. You can call them using a syntax similar to TurboIMAGE intrinsics. However, there are more OMNIDEX intrinsics than TurboIMAGE intrinsics. Likewise, OMNIDEX intrinsics perform more functions than TurboIMAGE intrinsics.

OMNIDEX intrinsics are subdivided into the following categories.

- ❑ OMNIDEX (ODX) keyword access intrinsics perform keyword retrievals (searches against keyword keys) and their related functions, like transferring data from qualified records.
- ❑ IMSAM (DBI) sorted access intrinsics perform sorted retrievals.
- ❑ IMSAM (DBI) general intrinsics perform basic OMNIDEX functions, including updating OMNIDEX indexes, and handling OMNIDEX-related error conditions and messages.

More detailed information about OMNIDEX intrinsics is available in the *OMNIDEX ImagePlus SDK API Guide*.

Programming considerations

While the OMNIDEX general intrinsics are similar to TurboIMAGE intrinsics, there are a few fundamental differences. They are discussed next.

Parameters

OMNIDEX intrinsics use the same number and types of parameters as their corresponding TurboIMAGE intrinsics. They execute identically to their TurboIMAGE counterpart if a standard TurboIMAGE mode value is used. This means that a program where TurboIMAGE intrinsics are replaced by OMNIDEX intrinsics (with no other changes) operates identically to the TurboIMAGE version. The only differences are:

Sorted access and general intrinsics accept several additional values for the *mode* parameter.

For example, DBIGET uses the mode values 90, 91, and 92 to read through the key values indexed for a sorted key. DBIINFO supports all the TurboIMAGE mode values, plus additional mode values that return information about how OMNIDEX keys are installed on a database.

DBIEXPLAIN has an additional parameter called *parm*, which lets you configure how errors are handled.

The **DBIGET** *dset* parameter is slightly different from the **DBGET** *dset* parameter. It is 16 halfwords (32 bytes) long, and includes an 8 word (16 byte) value for the IMSAM key name.

OMNIDEX intrinsics require a 21 word *status* array instead of the 10 word array required in calls to TurboIMAGE intrinsics.

Words 1-10 are still used for the information returned by the TurboIMAGE call, word 11 is the OMNIDEX condition word, and words 12-21 vary according to the intrinsic being called, and the type of error handling being used.

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

For OMNIDEX intrinsics, the three types of mode options are described by their targets:

Normal mode	affects both the TurboIMAGE data sets and the OMNIDEX indexes, and is the default if no mode options are used.
IMAGE-only mode	affects only the TurboIMAGE data sets, and may be used for extensive batch updates.
Index-only mode	affects only the OMNIDEX indexes, and is used to maintain stand-alone B-trees, and to perform index-only retrievals.

4

Additional retrieval features

In addition to the search arguments and operations described on page 4-12 through page 4-17, the OMNIDEX Intrinsic Interface has other features that the Standard Interface to Third Party Indexing does not. They are discussed next.

Keyword-only searches (keyword look-ups)

You can use ODXFIND modes 10 or 11, followed by ODXGETWORD, to return a list of the indexed keyword values for any keyword key. This is called a keyword-only search or a keyword look-up. Keyword look-ups are useful in a variety of situations. They let you:

- ❑ give users an idea of what values they can use to search a given keyword key.
- ❑ assess the number of noise words (words that occur frequently and are useless for retrieval), which can be excluded from indexing to save disk space.
- ❑ check for misspelled keyword values.
- ❑ list numeric values and their distribution.

ODXFIND mode 10 or 11 finds all the keywords that match a generic (partial) keyword value or fall within the range of two keyword values. Mode 10 also returns the number of keywords qualified by the generic or range argument. Mode 11 returns only the keywords qualified by the argument. After finding keywords, use ODXGETWORD to retrieve them.

In keyword look-ups, the keyword list must contain only one generic keyword or a keyword range terminated by a semicolon (;) or space. For example, DAT@; and DAA TO EZZ; are acceptable arguments for keyword look-ups. DA??; is not an acceptable argument. The partial-keyword or keyword range determines which keywords that will be qualified.

After you've performed a keyword look-up, use ODXGETWORD to return the actual words (and the number of times they occur) sorted in alphanumeric order to the calling application. See the "Intrinsics" chapter of the *OMNIDEX ImagePlus SDK API Guide* for more information about ODXGETWORD.

Transferring Data

The ODXTRANSFER intrinsic transfers data from qualified records to files. This is useful when supplying arguments for a Multifind operation, creating subfiles for reporting purposes, and saving an internal ID list for future searches.

In DataView you can transfer IDs to a file using the */X* or *}filename* prompt window commands. This lets you save record IDs qualified from one or more previous calls to ODXFIND, and further qualify them later, in any number or combination of searches.

For detailed information about ODXTRANSFER, see the “Intrinsics” chapter of the *OMNIDEX ImagePlus SDK API Guide*.

External ID files

As mentioned above, you can save an internal ID list to a file. To reload the IDs in the file, pass them through the ODXFIND *keywords* parameter (the keyword list) by placing the file name, preceded by \$, (like *\$filename*). *\$filename* can be used like, and combined with, other keyword arguments.

When using an ID file in a search, you must be searching the same OMNIDEX domain where the IDs were qualified. If you use an ID file that was saved from a search on a different domain, there is no meaningful correspondence between the IDs in the file and the records in the table (data set) that you are searching. Similarly, after reindexing unlinked details, and domains whose master does not use an integer search item the IDs are reassigned and previously saved ID files are not valid.

Compression of detail searches

By default, keyword keys installed on detail data sets (tables) qualify individual detail records, regardless of their search item values or relationship to a master record. This is known as *record specific* retrieval. When a detail data set is linked to a master data set, you can convert a search that qualified individual detail records to qualify record complexes (master records and their linked detail chains). This is known as *list compression*.

To compress the results of a keyword search, call ODXFIND mode 30 after the detail records are qualified. Note however, that two conditions must be met.

- The keyword search must have been performed on a linked detail set.
- The keyword search must not have been performed against a Record Complex key.

Note also that after the compression is complete, the record specific qualifying count (*status* words 3-4) matches the record complex qualifying count (*status* words 5-6).

Index-only mode on sorted keys

In index-only mode, the requested key values are retrieved directly from the indexes without retrieving any TurboIMAGE records. Because index-only mode retrievals greatly reduce the number of disk accesses required for each retrieval, they deliver better performance.

To perform an index-only mode retrieval, add 1000 to the DBIGET *mode* value. The key values are returned to your calling application via the *argument* parameter. After the first matching key value is returned to your application, the application can call DBIGET to perform index-only sequential reads (mode 109*n*) to return additional indexed key values to the calling application.

Index-only retrievals are typically performed on composite sorted keys that contain all the data required by the calling application. See "Designing composite keys", on page 4-43, for information about designing composite sorted keys for index-only mode.

The Standard Interface to Third Party Indexing

The Standard Interface to Third Party Indexing is available for TurboIMAGE versions C.04.03 and later, or MPE/iX version 4.0 or later. The primary difference between the TurboIMAGE/iX intrinsics and any previously released TurboIMAGE intrinsics is that the newer intrinsics use additional *mode* values to support keyword and sorted access. When these modes are used, TurboIMAGE calls the OMNIDEX intrinsics in XLOMNIDX.PUB.SYS to perform sorted or keyword key retrievals.

The Standard Interface to Third Party Indexing automatically updates OMNIDEX indexes as it updates TurboIMAGE data. This means that you can use existing TurboIMAGE update applications through the Standard Interface to Third Party Indexing without changing them or referencing additional libraries. You must, however, enable your databases for Third Party Indexing, as discussed in "Updating OMNIDEX Data", on page 4-50.

Enabling OMNIDEX searches

For DBFIND mode 1, if the *item* parameter references an OMNIDEX key, DBFIND automatically uses the OMNIDEX indexes to find records. However, when the *item* referenced in a call to DBFIND mode 1 is a TurboIMAGE key or a sorted key, as well as an OMNIDEX key, the type of search performed in DBFIND mode 1 can vary based on the *argument*. To be sure an application performs either a keyword search or sorted search, your applications should call the specialized TPI modes added to DBFIND. See the *OMNIDEX ImagePlus SDK API Guide* for information about developing TurboIMAGE retrieval applications.

Additional retrieval features

In addition to the search arguments and operations described on pages 4-12 through 4-15, the Standard Interface to Third Party Indexing has other features that the OMNIDEX Intrinsic Interface does not. They are discussed next.

Sorted key retrieval capabilities

Sorted key retrievals in the Standard Interface to Third Party Indexing are generally less complicated than in the OMNIDEX Intrinsic Interface. Ranges, for example, are supported by Standard Interface to Third Party Indexing intrinsics. Instead of having to programmatically check for a start or stop value, as in the OMNIDEX Intrinsic Interface, searches performed using the Standard Interface to Third Party Indexing define the boundaries of the retrieval through the DBFIND *argument* parameter. Subsequent calls to DBGET return a “beginning of chain” or “end of chain” condition when the range boundaries are reached. The Standard Interface to Third Party Indexing can also return a count of the number of records that qualify for a sorted retrieval.

With the exception of range operations on binary sorted keys (DBFIND mode 11), all sorted search operations begin with a call to DBFIND mode 1. Relational and range operations are specified by the use of tokens, as discussed on page 4-17.

Counts for sorted retrievals

In addition to providing counts of qualified records for keyword key searches, the Standard Interface to Third Party Indexing also provides a count of records qualified in a sorted search. This differs from the OMNIDEX Intrinsic Interface, which does not return a count for sorted retrievals. Because counting sorted key values requires extra overhead, this feature is optional. See the "DBFIND" section of the "Intrinsics" chapter in the *OMNIDEX ImagePlus SDK API Guide*.

Pattern matched and generic retrievals

DBFIND supports pattern matching for searches on character type sorted keys, as described on page 4-7. For example, if you use the argument `??????` against an X12 sorted key, it qualifies all records whose full key value contains only six ASCII characters. The argument `#####` qualifies all records whose full key value contains only six digits.

Relational operations

DBFIND mode 1 supports relational operations for sorted keys. To perform a relational operation on a sorted key, simply precede the full, partial, or pattern-matched key value with a relational operator (`>`, `<`, `<=`, `>=`).

Note that certain restrictions apply when using pattern-matched arguments in relational operations. See "Pattern-matched and generic arguments" on page 4-7 for more information.

ASCII ranges

Ranges on ASCII sorted keys are supported by DBFIND mode 1 in either of two ways.

- ❑ You can use the range syntax *startvalue* TO *stopvalue*. This qualifies all records with key values between the *startvalue* and *stopvalue* inclusive. You can also use : (a colon) as a range operator.
- ❑ You can use two relational expressions. For example, *>startvalue* *<stopvalue*. This lets you perform exclusive ranges (where the *startvalue* and *stopvalue* are not included).

Binary ranges

A separate *mode* value, mode 11, supports range operations for binary or ASCII values. No range operator is used. Instead, the starting value and stopping value each use one half of the *argument* parameter. Therefore, the *argument* parameter must be large enough to hold two binary search arguments.

You can simulate relational operations for binary values by supplying only a starting value ("greater than or equal to") or only a stopping value ("less than or equal to").

Determining Retrieval Requirements

This section can help you decide how to install keyword and sorted keys on a database. It is divided into five sections.

- ❑ “Ordered data vs. free-form data”, on page 4-29, discusses what keys to install on fields, based on the data stored in a field.
- ❑ “Which fields make good keys?”, on page 4-30, discusses how to use a field’s attributes to determine what type of key to install on it.
- ❑ “Reports as indicators”, on page 4-33, discusses how record selection and report performance can indicate the need for keyword or sorted keys.
- ❑ “Retrieval possibilities”, on page 4-36, discusses how user needs can determine what type of keys to install.
- ❑ “Designing composite keys”, on page 4-43, discusses how to combine or isolate fields or parts of fields into composite keys.

Noting keys for installation

As you read this section, and retrieval needs become clear, you might note them on a copy of your database’s schema. These notes can help you during installation.

Some notations you might make next to a field are:

- ❑ KW (to designate it as a keyword key)
- ❑ SK (to designate it as a sorted key)

If you do not have a schema on hand, you can generate one using DBMGR’s SCHEMA command as discussed next.



You must have exclusive access to a database to generate a schema using DBMGR.

1. Log on as the database creator or the database account manager and run DBMGR:

```
RUN DBMGR.PUB.DISC
```

2. When prompted, enter the database name:

```
Database name: dbname.group
```

3. At the DBMGR: command prompt, enter SCH, for SCHEMA:

```
DBMGR: SCH;NOPASS
```

Appending NOPASS insures that, for security reasons, the password doesn't print.

Finally, you are prompted for a destination. Enter LP to print the schema to the system line printer.

```
Generate a schema for the current database
```

```
Destination: LP, filename or <cr>? LP
```

Ordered data vs. free-form data

In TurboIMAGE, key data is usually stored in a predictable order to make each record accessible. Other non-key fields can store data in a free-form manner. In TurboIMAGE, free-form data mostly provides information about a record. For example, an ADDRESS field may provide information about an individual CUSTOMERS record, but it would not make a good key because of the free-form data it contains.

A major factor in choosing a key type (keyword, sorted, or both) for a field is whether the field contains ordered or free-form data. A field's function also can help determine what type of key to install. The factors involved in selecting fields for key installation, and determining what type of key to install, are discussed below.

Ordered data

As mentioned above, TurboIMAGE key fields typically store data in a predictable order. Some examples of ordered fields are:

- search item (SI) fields
- code fields
- hierarchical fields
- date fields

Sorted keys are best installed on ordered fields because they return records sorted in order by key values. Ordered fields also support sorted partial-key retrievals, especially if their data is ordered hierarchically.

Free-form data

Name fields, like CUSTOMER-NAME, and description fields, like PART-DESC, which store information about the record, are often free-form. They do not depend on any predictable structure other than the field's data type and size. Textual or descriptive fields that contain names, comments or abstracts are good examples of free-form fields.

Fields that store free-form data are good candidates for keyword keys. Keyword access can make free-form fields instantly accessible by their contents when they are installed as keyword keys. This eliminates the need for serial reads, and speeds searches for free-form data.

The ordered or free-form nature of a field is only one factor in its suitability to keyword or sorted access. The other factors that determine which fields make good candidates for key installation are discussed next.

Which fields make good keys?

This section discusses which TurboIMAGE fields are good candidates for a keyword key, sorted key, or both. The effect of TurboIMAGE data types on OMNIDEX key installation is discussed in Appendix A.

Fields used in KSAM files

You can create sorted keys from fields that are duplicated in KSAM files. Sorted keys provide all of the retrieval and sorting capabilities of KSAM, but with greater security, greater flexibility, less redundancy, and faster index loading.

If several fields are used in a KSAM file, create a composite sorted key to replace that file. “Designing composite keys”, on page 4-43, contains more information about composite sorted keys.

Key fields

TurboIMAGE keys in details may be good candidates for OMNIDEX keys. Which type of key to install depends on whether the field is used to sort data, or whether the field is used with other fields to find records.

To support searches across domains (Multifind), you may install a keyword key on the TurboIMAGE search item of a master set¹ or an unlinked detail set. See “Multifind”, on page 4-63, for more information about performing Multifind searches.

Keys with paths to automatic masters are also good candidates for OMNIDEX keys. If the key defines a sorted TurboIMAGE path, you can replace it with a composite sorted key that comprises the TurboIMAGE key and its sort field. This provides sorted retrievals without the overhead associated with maintaining TurboIMAGE sorted paths.

If a field is used in combination with other fields to find records, install keyword keys on the fields and eliminate their automatic masters.

4

Textual and descriptive fields

Textual fields are free-form fields and, therefore, good candidates for keyword keys. In a PRODUCTS master, for example, PRODUCT-DESC contains product descriptions. Even if you know that a keyword, like “TERMINAL”, occurs in a PRODUCT-DESC field, you could never predict exactly where in the field that value might be found.

With keyword keys, you can locate records no matter where a keyword occurs in a field. Making PRODUCT-DESC a keyword key gives you keyword access to all records with the value “TERMINAL”, for example, in the PRODUCT-DESC field. You could even use ANSI,TERMINAL to find all product records for ANSI terminals.

-
1. When a keyword key is installed on the search item of a master set, the No Parse option is automatically installed on that key.

Hierarchical fields

Hierarchical fields are ordered fields. The initial (most significant) bytes of the field represent the most generic information; trailing bytes (least significant) grow increasingly more specific. General ledger account numbers, for example, are typically hierarchical. The initial two bytes might represent a DEPT code, the next four a BRANCH code, the next four a VENDOR code, and so on.

Sorted access is ideally suited to hierarchical keys, because you can search using as much of the key value as you want. You could retrieve records starting at a particular department, or at a particular branch within a department. This lets you retrieve sorted records as specifically or as generically as desired.

You can create a composite sorted key to reorder the hierarchy of a field. This is discussed later in “Designing composite keys”, on page 4-43.

Because parts of hierarchical fields often contain codes that categorize records (like a year, or department code), hierarchical fields are good candidates for composite keyword keys. This lets you key any part of a hierarchical field. You can search interior bytes that contain codes, alone or with other keyword keys, to find records.

Code fields

Code fields are ordered fields. Fields that contain data like zip-codes, and SIC codes lend themselves to both keyword and sorted access.

Installing a sorted key on a ZIP-CODE field lets you generate presorted mailing labels for bulk mailings.

Installing a keyword key on a ZIP-CODE field lets you qualify locations by ZIP-CODE in conjunction with other free-form fields. For example, a user may want to find records based on both the ZIP-CODE and a DESCRIPTION field.

Many code fields are hierarchical. You can reorder long code fields, or divide them, into composite keyword or sorted keys. This can provide better access to the data they contain. See “Hierarchical fields”, on page 4-32, and “Searching parts and combinations of fields”, on page 4-38, for more information.

Date fields

Date fields are predictable, because you know the order in which the data is stored. Dates stored hierarchically in YYMMDD order are the most useful, because they support partial-key searches by year and month. If a date field is stored in MMDDYY order, you can reorder it by creating a composite sorted key or composite keyword key.

Proprietary date fields (used by ASK and PowerHouse for example) can pose problems to other non-proprietary applications. A keyword key installed on proprietary date fields lets you search proprietary date fields using conventional date formats with no programmatic conversion!

To retrieve particular records by date alone, install a sorted key on the date field of that set. To retrieve records sorted by date and other fields, create a composite sorted key that combines the date field with the other search/sort fields. To retrieve records by date and other fields in unpredictable combinations, install keyword keys on the date field and other potential search fields of that set.

When installing keyword keys on fields that contain ASKDATE, or PowerHouse JDATE or PHDATE values, be sure that the field is of data type K. If it is not, use the Type Casting option to index the key as type K. See page 2-11 of the "OMNIDEX Installation" chapter for more information about the Type Casting option.

4

Reports as indicators

One of the first places to look for fields that might benefit from OMNIDEX is in reports that cause performance problems. When you examine such reports, you should consider the following issues.

- How is the data retrieved and listed? (Multiple criteria? Sorted order?)
- How many records are retrieved?
- How much time (and disk access) is involved per report?

If you talk to the users who generate the reports, you also may discover other issues.

- What additional retrieval capabilities are desired?
- What reports are better suited to online applications?

Some considerations to help you decide how to install OMNIDEX on your TurboIMAGE database are discussed next.

OMNIDEX retrievals or serial reads?

Some reports take a very long time. This typically happens when an application relies on serial reads to select the qualifying records. Serial reads require many disk accesses (I/Os). Although great progress has been made in processor speeds, the best performance one can expect from the average disk drive is about 30 I/Os per second. Therefore, database applications are often disk-I/O-bound.

Take, for example, a report that lists all of the CUSTOMERS in Colorado. Suppose that of 100,000 records, only 500 qualify. Suppose also that there is a 20% pad space to allow for efficient hashing, so that the capacity of CUSTOMERS is 120,000. A reasonable blocking factor for most manual master sets is approximately six (6).

To qualify the 500 CUSTOMERS in Colorado, DBGET serially reads the entire data set (including vacant addresses). Because it searches blocks of records, the total number of I/Os equals the capacity (120,000) divided by the blocking factor of six (6). The total number of I/Os needed to locate the records is 20,000. If the disk access time is 30 I/Os per second, our search takes 666 seconds, or 11 minutes.

Because OMNIDEX searches indexes for record identifiers (IDs), the time required for a search depends on the number of records that qualify and your CPU class. OMNIDEX qualifies between 40,000 and 225,000 records per second. To qualify the 500 CUSTOMERS in Colorado would take about one-hundredth of a second.

DBIGET or DBGET also uses one I/O to retrieve each record, bringing the total number of I/Os to a fraction more than 500. If the disk access time is 30 I/Os per second, our search takes about 17 seconds. This is 39 times faster than a serial read.

If our sample data set had a capacity of 1,200,000 records, you can see that OMNIDEX would reduce your retrieval I/Os, and time, by one order of magnitude.

A good rule to determine if your report could benefit from a keyword key is: **If a search usually retrieves a percentage of records less than one third the inverse of the data set's blocking factor (here, 1/18, or roughly 6%), then keyword keys provide the most efficient means of retrieval.**

Check the input area of the report procedure files that take the longest. Look for conditional search statements that reference non-key fields. The fields they reference may be good candidates for keyword keys. This is especially true if the report selects records based on several fields or several conditional values.

Reports to convert to online

Often, a report can benefit from being converted to an online application. This is typically the case when:

- you frequently change the criteria used to select the records (*ad hoc* requests for data).
- a qualifying count is all that is needed.
- data is reported in sorted (hierarchical) order.

Frequent *ad hoc* requests

When the search criteria for a given report can vary each time it is run, the report is said to be *ad hoc*. Reports that prompt the user to input search values could be considered *ad hoc* reports. The speed and flexibility of OMNIDEX make it ideal for online, *ad hoc* applications.

If you have reports that prompt the user for search values, examine the source code file and see which fields or files are used for selection. You might consider selecting those fields for keyword access.

When counts are all that is needed

Often, the number of qualifying records (the qualifying count) is all of the data your users need. For example, "How many orders were placed for a given product during a given month?" or, "How many orders were taken by a given customer service representative?" Look for reports that generate only totals of records.

Because keyword searches always return a qualifying count of records, keyword keys are ideal for applications that require only counts. Examine the source code files of such reports. See which items are used for selection, and consider installing keyword keys on those fields.

Reports that sort data

Some reports are generated by selecting all records, sorting them, and listing them in sorted order. Some examples of this are general ledger reports and alphabetic reference lists.

By installing a sorted key on the field used for sorting, users can review records online, in sorted order. They can even browse forward and backward through the sorted records. If you want to sort records by more than one field, you can create a composite sorted key that incorporates several sort fields.

Other reports select and sort large amounts of data, but report only summary totals. Some examples of this are general ledger summary and sales-by-territory reports. Whether data is listed to a terminal or to a line printer, sorted keys can help. Consider installing a composite sorted key composed of the sort fields and the amount being summarized. This allows for extremely fast index-only mode retrievals. Because the required data is stored in the sorted key's index, there is no need to retrieve records from the data set. See the *OMNIDEX ImagePlus SDK API Guide* for a detailed discussion of index-only mode retrievals.

Retrieval possibilities

Although reports can benefit from OMNIDEX, its speed makes it suitable for many online applications. *The IMAGE/3000 Handbook* (Wordware) sets a target benchmark of ten seconds per search for an effective online application. OMNIDEX typically provides retrieval times well within ten seconds, even during heavy system activity.

OMNIDEX's flexibility supports many previously impossible retrievals. This section explores many of these retrieval possibilities.

Combining keyword arguments

OMNIDEX lets you find records by combining several keyword arguments against a given keyword key. You can combine the keywords in a variety of expressions including:

- ❑ Boolean expressions (AND, OR, and NOT), like (SOFTWARE AND CONSULT@) OR CONTRACT@ against a DESCRIPTION key field
- ❑ relational expressions (>, >=, =, <=, <), like >=10000 against a BALANCE key field
- ❑ ranges (: THRU TO), like 80200 TO 80499 against a ZIP-CODE key field

You can also convert date search values to the correct storage format automatically. This lets you search keyed ASKDATE and PHDATE fields without having to programmatically convert dates to a proprietary storage format during searches.

Searching several keys

OMNIDEX also lets you find records by using keyword arguments or expressions against several keyword keys. For example, you might search the COMPANY-NAME key field of a CUSTOMERS master using the keyword combination of HEWLETT AND PACKARD, and then search the STATE key field using the keyword argument NOT CO. This would return all CUSTOMERS records for Hewlett Packard offices that do not reside in the state of Colorado

To progressively qualify records using several ungrouped keyword keys, you must search each key in successive calls to the search intrinsic. In the example above, you would use HEWLETT PACKARD to search COMPANY-NAME; then, NOT CO to search STATE. Because OMNIDEX returns a qualifying count for each keyword search, you can see how many records qualify for each stage of a progressive search.

You can search several keyword keys in one intrinsic call if they are grouped. Grouped keys are treated as one logical key. In the above example, you could use the keyword combination HEWLETT AND PACKARD NOT CO if the COMPANY-NAME key was grouped with the STATE key. See "Grouping", on page 2-9.

Searching across data sets

OMNIDEX supports progressive searches on keys that reside in different sets, as long as the sets were linked to each other, or to the same master, during installation. See "Linking Details to Masters", on page 4-56, for more information.

For example, the CUSTOMER-NOTES detail is linked to the CUSTOMERS master. If STATE is a keyword key in CUSTOMERS, and ACTION is a keyword key in CUSTOMER-NOTES, you could qualify all CUSTOMERS master records for the STATE of Illinois whose CUSTOMER-NOTES showed an ACTION pertaining to "INTEREX". When searching on CUSTOMER-NOTES, you could choose whether you wanted to return individual detail records or entire detail chains.

You can perform searches across sets simultaneously or successively. If keys are grouped across linked sets, you can enter keyword arguments for both keys in one intrinsic call. If the keys are not grouped, then you must enter the arguments in successive intrinsic calls. The rules are the same as discussed in "Searching several keys", above.

OMNIDEX also provides the ability to search across sets that were not linked during installation. This can include sets that reside in different databases. You must use Multifind to do this, as discussed on page 4-63.

Searching parts and combinations of fields

OMNIDEX lets you key parts and combinations of fields, as well as combinations of parts of fields. Such keys are called composite keys. You can create composite keyword keys and composite sorted keys.

Composite keyword keys are useful for:

- accessing parts of fields
- rearranging the order of a field
- accessing individual elements of TurboIMAGE arrays

Composite sorted keys are useful for:

- sorting records by several fields
- accessing parts of fields
- rearranging the order of a field
- accessing individual elements of TurboIMAGE arrays

Designing composite keyword and sorted keys is discussed in “Designing composite keys”, on page 4-43.

Examples of composite key applications are listed below.

Combining fields

Combining fields into a composite sorted key lets you search for records by several fields, as well as sort records by several fields. For example, to find (and sort) ORDER-LINES records by their product numbers and the dates of shipment, you could combine the PRODUCT-NO and DELIVERY-DATE fields, in that order, into a composite sorted key (like PROD-DD).

Rearranging fields

Rearranging the order of a field is often useful. For example, you could create a composite key (keyword or sorted) to rearrange a date field from MMDDYY order, into a hierarchical YYMMDD order. This composite key would support partial-keyword arguments, like 8901, relational retrievals, like >=8901 (or >=8901@), and range retrievals, like 890201 TO 890215.

Rearranging hierarchical fields into a different sort order through a composite sorted key is often useful in report applications. Take, for example, an ACCOUNT-NO key that contains a DEPT code, followed by a BRANCH code, followed by a VENDOR code. To sort records by VENDOR first and DEPT second, you could create a composite sorted key where the VENDOR code comes before the DEPT code.

Keying parts of fields

Keying parts of fields can be useful if, for example, you need to sort or retrieve records based on the last four bytes of a character field. In composite sorted keys, parts of fields are often combined, as discussed above.

Keying parts of fields is useful for keyword searches on individual subfields that compose an ASCII field. Take, for example, the ACCOUNT-NO key discussed above. By creating three composite keyword keys you can gain keyword access to any or all of the DEPT, BRANCH, and VENDOR subitems.

Keying elements of TurboIMAGE arrays

Keying elements of TurboIMAGE arrays lets you search any element of an array, regardless of its position. This is similar to keying parts of fields, described above.

Many databases use arrays — also known as compound items — to store data. For example, an ACCOUNT-SUMMARY data set (table) may keep totals for the twelve months of a year in a compound item of TurboIMAGE type 12P12 called ACCT-BAL. This compound item is a TurboIMAGE array consisting of twelve packed decimal items, each 12 nibbles (6 bytes) long.

To install a sorted key on the third element of ACCT-BAL, you would create a new key with a new name (like MARCH) defined as bytes 13 through 18 of ACCT-BAL. When you had defined its single component, you would assign a sorted key type to the key. The actual process is described in “Creating composite keys”, on page 2-4.

By default, when you install a keyword key on a compound item, OmniUtil treats it as several elements combined into one logical key, much like grouped keys. Keywords are parsed according to element boundaries, as well as the spaces and special characters that separate them.

To index an array as though it was one field, use the Blob Indexing option, as discussed on page 2-8. This is useful when words in an array span element boundaries.

To install a keyword key on an individual element of an array, the fifth element of a 12P12 ACCT-BAL array for example, you must create a composite keyword key with one component defined to start at byte 25 for a length of 6. Because OmniUtil treats composite keys as ASCII by default, you must also typecast the field as P. See “Type Casting”, on page 2-11, for more information.

Data retrieved

The type of data you want to retrieve can help determine whether to install a keyword or sorted key on any field. Each retrieval category below is described with respect to the type of indexing (keyword or sorted) it requires. Sometimes both keys are appropriate for a field. You can install both types of keys on any field.

Retrieving records

You can use both keyword and sorted keys to retrieve individual records or to list fields of qualifying records.

Sorted access retrieves records in sorted order by one or more fields, based on full or partial-key arguments. Sorted keys let you sort records by a particular key value. Sorted composite keys can incorporate several fields and let you sort records by several fields.

Keyword access qualifies a subset of records based on keyword values. For this reason, keyword keys are best suited for retrieving a small subset of records based on simultaneous or successive searches.

Retrieving key values

There are differences between the way OMNIDEX retrieves keyword key values and sorted key values.

You can retrieve the key values for any keyword key and display them in sorted order, with the number of records that contain each keyword. This is called a keyword-only search or a keyword look-up. The ability to list keyword values lets you:

- provide users with an idea of what values they can use to search a given keyword key
- assess the number of noise words (words that occur frequently and are useless for retrieval). These can be excluded from keyword indexing to save disk space
- check for misspelled keyword values
- list numeric values and their distribution

You also can retrieve sorted key values only. Sorted key values alone can be a valuable source of data. Some applications use long, hierarchical composite sorted keys that can store an entire logical record.

Index-only mode is the term applied to sorted retrievals that return the key value only. Index-only retrievals are used for high speed data processing. For more information about index-only mode, see the *OMNIDEX ImagePlus SDK API Guide*.

Sorting records

Sorted access to records is available through sorted keys. By installing a sorted key on a field, you can sort records in ascending or descending order by their values for that field.

To sort records by more than one field, create a composite sorted key. Composite sorted keys can isolate or combine fields or parts of fields in the same data set. Key values (and pointers to the records that contain them) are sorted by the key's component fields, in the order that components were specified during installation. Note any fields you may want to combine into a composite key. See "Designing composite keys", on page 4-43, for more information.

Qualifying count

Keyword key searches return a count of the number of records that qualify in keyword searches. This count can either be:

- the number of SIs for masters, and for linked details that contain Record Complex keyword keys
- or
- the number of relative record numbers for all other details

Record complexes and record specific detail retrievals are discussed in “Linking Details to Masters”, on page 4-56.

Designing composite keys

This section discusses how to create composite keyword or sorted keys from physical fields. As you decide which data sets could benefit from the creation of a composite keyword (KC) or sorted key (SC), you can note on your schema which fields to combine into which composite keys. This can help you when you install OMNIDEX.

If you are creating a composite key from parts of fields, OmniUtil requires you to specify a start byte and byte length for each component. To denote a part of a field, use the syntax of *[startbyte:bytlength]*. An X10 field where bytes three through six comprise component 2 of a composite sorted key should be noted as *fieldname [3:4] Component 2 of keyname-SC*.



Composite keys can only combine fields within the same data set.

Some sample notations you might make on your schema for a data set are:

- AREACODE-KC (composite keyword key)
Component 1: PHONE [1:3]
- SALES-MONTH-SC (composite sorted key)
Component 1: SALESMAN
Component 2: LAST-PUR-DATE [1:4]

The order that you specify components is important for composite sorted keys. This is discussed later, on page 4-46.

Composite keyword keys

Certain rules apply when designing and creating composite keyword keys.

- ❑ Composite keyword keys cannot be created for an automatic master set.
- ❑ The maximum length of a composite keyword key is 255 bytes.
- ❑ The maximum number of components for a composite keyword key is 250.
- ❑ Composite keyword key components must be from within the same data set.
- ❑ Composite keyword keys can only combine character (type U and X) components. Composite keyword keys created from type I, J, K, P, R, or E items can contain only one component.
- ❑ Composite keyword keys are indexed as ASCII by default. To index them as another data type, you must typecast them.
- ❑ If a composite key is typecast, and the type specified is not X, then only one component is accepted.

You can use composite keyword keys to combine fields or to rearrange them as discussed next.

Combining fields

Any number of components can be combined in a composite keyword key, if the components reside in the same data set, and the length of the composite key does not exceed 250 bytes.

Combining fields is useful for finding individual detail records based on combinations of values from several fields, as described in “Linking Details to Masters”, on page 4-56. This is most useful when the fields you want to search are also installed as Record Complex (RC) keyword keys. Composite keyword keys ensure that all criteria for the combined fields are met in the same detail record, and return a count of the number of individual detail records that qualified.

Rearranging fields

Composite keyword keys are also useful for reordering bytes of fields. For example, you can use a composite keyword key to rearrange a date field from MMDDYY order into YYMMDD order. This would better support partial retrievals like 8901@, and range retrievals, like 890201:890215.

Keying parts of fields

Composite keyword keys can help you find records based on varying combinations of subitems from an array or hierarchical field.

Take, for example, a DEPARTMENT key that contains three subitems: a DEPT code, a BUREAU code, and an OFFICE code. You could create a composite keyword key for each subitem to support retrievals on any or all of them, independently or in combination.

In the case of an integer array (like a 4J2 array), you can isolate individual elements into composite keyword keys. For integer arrays, you must typecast each composite keyword key as type J. See "Type Casting", on page 2-11 for more information.

Composite sorted keys

Composite sorted keys let you create sorted keys from parts of physical TurboIMAGE fields in a given data set. You can combine fields for partial chained reads and sorted access across several fields in the same data set.

You can also create a composite key to isolate part of a field, or reorder a field. For example, if you had a date field in DDMMYY order, you could create a composite sorted key for YYMMDD, or simply YYMM.

Composite keys may be composed of any fields or portions of fields within a data set, including TurboIMAGE search items. The following limits apply to composite sorted keys.

- ❑ The maximum length for a composite sorted key is 250 bytes.
- ❑ The maximum number of components for a composite sorted key is 250.
- ❑ When creating a composite sorted key, the most frequently searched field should be the first component of the key. For example, to qualify records by STATE, and sort them by STATE and DATE, create a key with the STATE field as Component 1, and the DATE field as Component 2. To qualify records by DATE, and sort them by DATE and STATE, create another composite key with DATE as Component 1.
- ❑ When creating a composite sorted key to sort records by two or more fields, specify the component fields in the order that you want the records sorted. For example, to find records by PRODUCT-NO and to sort the returned records by their DATE and STATE fields, create a composite key with PRODUCT-NO as Component 1, DATE as Component 2, and STATE as Component 3.
- ❑ You can create a sorted composite key that combines components of different data types (such as an X4 and J2) to sort records by several fields. However, if you try to supply arguments for both components, the results are unpredictable.

Some applications of composite sorted keys are described next.

Sorting on several fields

A composite sorted key comprised of more than one field lets you retrieve records in sorted order by two or more fields.

For example, you could create a composite sorted key that combined the PRODUCT-NO and DELIVERY-DATE fields in an ORDER-LINES detail set. That way, you could retrieve records by product number, and return the qualifying ORDER-LINES records sorted by DELIVERY-DATE within PRODUCT-NO.

Creating hierarchical keys

You also can create hierarchical composite keys by combining two or more fields. A partial-key search on such a key would return only the records for a level in the hierarchy.

Partial chained reads

Composite sorted keys can also combine a TurboIMAGE SI with another field in a detail data set, so that you can start retrieving from anywhere along the chain. Such a key lets you perform a partial chained read, reading only the portion of the chain that you need to access. This read is much faster than a normal chained read.

For example, a composite sorted key, CUST-DATE, could combine of CUSTOMER-NO, the SI of the ORDER-LINES detail set, with the DATE-ENTERED field. A partial-key search on this composite sorted key, could find the ORDER-LINES records for a particular customer on a particular date. For example, using the argument 1088@, you could list all records for customer number 10 in 1988.

Eliminating sorted paths

Composite sorted keys can replace sorted paths, which are costly in terms of update overhead, by combining the search item and sort fields of a set. For example, you could combine PRODUCT-NO and DELIVERY-DATE of the ORDER-LINES detail set into a composite sorted key named PROD-DD. You could retrieve product orders in sequence by product number and date, without maintaining a sort path on DELIVERY-DATE.

Because sorted keys support relational operations (<, >=, =, >, <=), and ranges, you can easily report the deliveries of products over a given period.

Several composite sorted keys may increase report performance if the primary search or sort fields differ from one report to the next. If several reports on a given data set each sort records by different fields, you may want to define other composite sorted keys. This eliminates having to sort the records through the report application.

Index-only mode retrievals

Besides their ability to sort data, a strong advantage to using composite sorted keys is the ability to perform index-only retrievals. Normally, OMNIDEX applications search the indexes for SIs or relative record numbers. Then, they call DBGET to retrieve the qualifying records from your TurboIMAGE database.

In index-only mode, the requested key values are retrieved directly from the indexes without calling DBGET for any TurboIMAGE records. Because index-only retrievals greatly reduce the number of disk accesses required for each retrieval, they deliver better performance.

To support index-only retrievals, include all reported items in the composite sorted key. Specify the items that you would search to qualify records as the leading components of the key. This increases the efficiency of searches and offers better performance.

**NOTE**

The only limitation to using composite sorted keys for index-only retrievals is that they can't be longer than 250 bytes, including the length of the TurboIMAGE search item, for master sets, or the relative record number, which is 4 bytes, for detail sets.

Comparison of keyword and sorted keys

Attribute	Keyword Keys	Sorted Keys
Ideal field type	Ideal for textual (character) information like names and addresses or for codes and data with predictable formats.	Ideal for data with predictable positions like dates or any field where sorted access (either alphabetical or numerical) is desired.
Indexing capabilities	Indexes each word or value in a field separately (unless the NO PARSE option is specified).	Indexes the key field as a whole. (Keys that index part of a field can be created.)
Retrievals	Supports retrieval by a word or value anywhere in a field.	Supports retrieval by a key value that must start with the leftmost characters or values in the field.
Partial-key retrieval	Supports partial (generic) keywords via an at-sign (@).	Supports partial-keys.
Range and relational retrievals	Supports ranges of generic and fully specified keywords. Supports relational retrievals via keyword ranges and the relational operators >, >=, <, <=, and = for generic and fully specified keywords.	Ranges are supported by the Standard Interface to Third Party Indexing, or programmatically through the OMNIDEX Intrinsic Interface. Relational retrievals supported via relational modes, or >, >=, <, <=, and =.
Boolean retrievals	Supports Boolean operators (AND OR NOT) in any combination for a keyword retrieval.	Simulates a Boolean AND operation via composite keys.
Qualifying count	Returns the number of qualifying records or record complexes.	Retrieves the first record that qualifies. Qualifying count available through Standard Interface to Third Party Indexing.
Qualifying lists	Qualifies records or record complexes. Finds all of the records or record complexes that meet the criteria.	Finds the first record that meets the criteria (matches the key or partial-key). Retrieves one record at a time.
Sorting	Retrieves records in order by the OMNIDEX ID.	Retrieves records in sorted-sequential order by the key.
Multiple field retrieval	Supports retrieval based on keywords for more than one field (multiple criteria). Also supports retrieval across sets and domains.	Supports retrieval based on only one key. (Composite keys, however, can combine values from more than one field.)

Table 4-7: Keyword and sorted keys compared

Updating OMNIDEX Data

When you update data in an OMNIDEX database you must also update the OMNIDEX indexes. As values are added to, removed from, or changed in the fields in a record, the key values indexed for that record must also be updated. If they are not, OMNIDEX keys do not reflect the most recent data, and can return false results when used in searches.

OMNIDEX provides intrinsics and utilities that update the data in indexes whenever data in keyed fields is updated. The greatest consideration you must make is when to update the indexes.

You can either update the indexes automatically when you update TurboIMAGE data, or you can update the indexes later, by reloading them. Both index updating options are discussed in the next sections.

Real time updates

By default, whenever you update your data with programs that reference XL.PUB.DISC, any corresponding OMNIDEX indexes are updated in real time. If TPI is not enabled for a database, you can disable real time updates by using special IMAGE-only *mode* values in the ImagePlus API, as discussed in the *OMNIDEX ImagePlus SDK API Guide*.



Updating OMNIDEX indexes in real time is not efficient when you are adding, updating, or deleting a high volume of records. It is more efficient to update the indexes in batch after a high volume of update transactions.

If your update programs directly call the OMNIDEX intrinsics DBIPUT, DBDELETE, and DBIUPDATE, you should either link such programs to XLOMNIDX.PUB.SYS, or reference XLOMNIDX.PUB.SYS at run time. DBIPUT, DBDELETE, and DBIUPDATE automatically update any corresponding OMNIDEX indexes after updating TurboIMAGE data.

Automatic indexing through TPI

If your programs use the Standard Interface to Third Party Indexing and call the TurboIMAGE intrinsics in XL.PUB.SYS, you may want to enable your database for Third Party Indexing, as described on page 4-84, to automatically update the indexes in real time. If a database is not enabled for Third Party Indexing, the indexes are not updated in real time, unless your update programs run through Call Conversion, as discussed next.

Automatic indexing through Call Conversion

As mentioned earlier, Call Conversion is a feature of OMNIDEX that lets you use your current update applications to update the OMNIDEX indexes in real time. If your database is not enabled for TPI, and you do not change your programs to call the OMNIDEX update intrinsics directly, then you should use Call Conversion to update OMNIDEX indexes in real time.

The Call Conversion library contains executable routines that lock and update the OMNIDEX indexes when you add, delete, or update key data in TurboIMAGE fields. The Call Conversion library traps TurboIMAGE update intrinsic calls and calls the appropriate OMNIDEX intrinsics to lock and update the OMNIDEX indexes. This allows the OMNIDEX indexes to be maintained automatically by TurboIMAGE update applications.

The Call Conversion library resides in XL.PUB.DISC. To use Call Conversion, your update applications (like Query, QTP, and SUPRTOOL) must resolve through the Call Conversion library.

You can do this in either of two ways.

- ❑ For native mode programs, reference the Call Conversion and OMNIDEX Intrinsic library routines in an XL= path, either at link time or at run time. For example:

```
RUN program.group.account,XL="XL.group,XL.PUB.DISC"
```
- ❑ For either native mode or compatibility mode, move the Call Conversion routines into your application programs' PUB groups, and run your applications with LIB=P. Compatibility mode call conversion routines reside in SL.PUB.DISC and USL.PUB.DISC.

The steps for setting up the Call Conversion libraries for each version of OMNIDEX are discussed in Appendix C.

Calling TurboIMAGE Intrinsic within the OMNIDEX Intrinsic Interface

If you are using the OMNIDEX Intrinsic Interface, you can intermix explicit calls to OMNIDEX intrinsics with implicit calls made by programs running under Call Conversion. For example, if you add OMNIDEX intrinsic calls to a TurboIMAGE application, you can still run the program under Call Conversion.

Similarly, you may have a vendor-supplied program that supports calls to external procedures. You can write external procedures that call the OMNIDEX intrinsics directly and still run the program through Call Conversion. The program's TurboIMAGE calls are converted to OMNIDEX intrinsic calls by the Call Conversion library, while the user-written procedures call the OMNIDEX intrinsics directly, as needed.

If you mix calls to TurboIMAGE intrinsics with calls to OMNIDEX intrinsics, then modify the program's error handling routines to call the appropriate intrinsics (DBERROR for TurboIMAGE calls, DBIERROR for OMNIDEX calls). For example, if a program calls DBGET under Call Conversion, then it should call DBERROR or DBEXPLAIN if the call to DBGET fails. If a program calls DBIGET, then it should call DBIERROR or DBIEXPLAIN if the call to DBIGET fails.

Batch Updates

Because real time updating of OMNIDEX indexes requires system overhead, some administrators update the OMNIDEX indexes (reindex) at night, in batch.

Batch updating of indexes is especially useful when updating a high volume of data. As a general rule, when updating more than 25 percent of the records in a file or database, you should disable the real time updating of indexes.

There are several ways to avoid updating the OMNIDEX indexes in real time.

- ❑ Install Batch Indexing on OMNIDEX keys that you would like to disable for real time updating.
- ❑ Disable the database for Third Party Indexing, and update it using programs that call TurboIMAGE update intrinsics directly, without Call Conversion.
- ❑ Use the OMNIDEX Intrinsic Interface to develop programs that call DBIPUT, DBIDELETE and DBIUPDATE in IMAGE-only mode. See “Updating Data” in the “Programming” chapter of the *OMNIDEX ImagePlus SDK API Guide* for more information.

Because none of these methods update the indexes, they require less overhead than real time (normal mode) updates. To update the indexes, use OmniUtil’s Reindexing Options menu. You can reindex the updated records in a session, provided you have exclusive access to the OMNIDEX index files. Or, you can schedule a reindexing job to stream when no one is accessing the index files.

The three methods for disabling real time updates of indexes are discussed next.

Using the Batch Indexing option

The recommended way to disable the real time updating of OMNIDEX keys in a database is to install them with the Batch Indexing option, as described on page 2-14.

The batch indexing option disables real time updating for individual keys. This lets you maintain some keys’ indexes in real time — those that must always reflect current data — while avoiding the overhead required to update all keys’ indexes. To maintain Batch Indexed keys’ indexes, you must periodically perform an OmniUtil indexing operation, as described in “Loading the Indexes”, on page 2-28.

Disabling Third Party Indexing for a database

You can avoid the real time updating of a database's indexes by your TurboIMAGE applications by not enabling, or disabling the database for Third Party Indexing. See "Real time updates", on page 4-50 to enable a database for Third Party Indexing.

When you use this method to update your TurboIMAGE data, you must consider the following:

- ❑ Disabling a database for Third Party Indexing does not disable real time index updates for all applications. Applications that resolve through Call Conversion (XL.PUB.DISC), and applications that directly call the OMNIDEX intrinsics in XLOMNIDX.PUB.SYS, may still update OMNIDEX indexes in real time².
- ❑ Disabling a database for Third Party Indexing affects the entire database. This can cause inconsistencies in online applications where keys must always reflect current data (like commodities market prices). You must reindex the database using OmniUtil to update the indexes to be current.

Disabling a database for Third Party Indexing involves the same steps outlined on page 4-84. The only difference is, when you are presented with the dialog box that says:



Select (Off). To synchronize the databases with your TurboIMAGE data, you must periodically perform an OmniUtil reindexing operation.

-
2. You can disable real time updating of indexes by using IMAGE-only mode, as discussed next.

Performing IMAGE-only updates

The OMNIDEX Intrinsic Interface update intrinsics provide mode options that let you update TurboIMAGE data without updating OMNIDEX indexes. Such updates are called *IMAGE-only* updates, and the mode used to enable them is called the IMAGE-only mode. This mode is available for DBIPUT, DBIUPDATE and DBDELETE by adding 100 to the *mode* parameter value when calling these intrinsics.

IMAGE-only mode is typically used when converting data from a TurboIMAGE database to an OMNIDEX-enhanced or IMSAM-enhanced database, or when performing large batch updates. After adding or updating data in IMAGE-only mode, reindex the affected data sets (tables) using OmniUtil.



NOTE

IMAGE-only mode does not work on TPI-enabled databases.

Linking Details to Masters

OMNIDEX uses TurboIMAGE paths between master data sets and detail data sets for reasons different from TurboIMAGE. The correspondence between masters and details primarily affects keyword keys.

This section discusses how linking — or not linking — a detail set to an OMNIDEX master affects keyword keys and keyword searches.

OMNIDEX masters

An OMNIDEX master is a manual or automatic master data set whose SI defines an OMNIDEX domain. It may contain keyword keys itself, or it may be linked to detail data sets that contain keyword keys. It is assigned either of two types of OMNIDEX IDs:

- ❑ If its SI is a 32-bit integer item (an I2, J2 or K2), it is assigned an SI/ID. That is, a record's SI value is its OMNIDEX ID value.
- ❑ If its SI is not a 32-bit integer item, it is assigned a transparent ID and an extra index is created to maintain a correspondence between each record's SI value and its internal OMNIDEX ID value.

OMNIDEX details

A detail data set that contains keyword keys is called an OMNIDEX detail. An OMNIDEX detail may be linked to an OMNIDEX master, or it can be indexed alone.

If the detail set is linked to an OMNIDEX master, their common SI is referred to as the *OMNIDEX SI*, and the detail is said to be a part of an *SI domain*. The procedure for linking details to masters is discussed in the "OMNIDEX Installation" chapter.

If the detail set is not linked to a master, it is said to be part of a *DR domain*, or a Detail Record indexed detail.

OMNIDEX domains

The American Heritage Dictionary defines a domain as: “1. A territory or range of control or rule. 2. A sphere of interest or action.” An OMNIDEX domain may be a single set, or several sets that share a common SI.

SI domains

An SI domain can link one master, and up to 16 detail sets, by a common search item (SI). Linking data sets into an SI domain enables a number of features.

- ❑ Multiple set retrieval qualifies records in one set and then further qualifies them in another related set. This concept is discussed earlier in “Retrieval possibilities”, on page 4-36.
- ❑ Grouping across sets treats several keys as one logical entity. For example, a COMMENTS field in a CUSTOMERS master may be grouped with a DESCRIPTION field in a CUSTOMER-NOTES detail during installation.
- ❑ Keywords for all of the sets in an SI domain are indexed in common indexes.

By default in version 3 of OMNIDEX, detail records from a set linked to an OMNIDEX master set are indexed individually by their relative record numbers and also as part of a record complex (see page 4-58) by their OMNIDEX SIs. Keyword keys that index detail records individually and by their master's SI are said to be *record specific (RS)*. This is different for earlier versions of OMNIDEX, where detail records that were linked to an OMNIDEX master were indexed only by their OMNIDEX SIs by default and were said to be *record complex (RC)*.

In terms of keyword searches, this means that whether or not a detail set is linked to a master, an ODXFIND search involving one or more keyword keys in a detail set returns only those detail records that qualify for all the combined searches. This concept is called *record specific indexing*, and is the default indexing for all keyword keys of detail sets that are linked to a master. After a keyword search on a detail set, to retrieve the SIs of qualified record complexes, instead of relative record numbers of individual detail records, call ODXFIND mode 30 to “compress” the qualified list.

If you want to always retrieve SIs after a keyword search, install the Record Complex option on keyword keys in detail sets.

Searching across data sets

Keyword keys in OMNIDEX master sets use SI values to locate records, while keyword keys in OMNIDEX detail sets use relative record numbers to locate records. Similarly, OMNIDEX maintains the correspondence between an OMNIDEX master and the detail sets in its domain by their common SI. Therefore, the relative records numbers returned by a keyword search in a detail set must be converted to their equivalent SI values before the search can progress to other sets within the same SI domain. This is discussed in “Compressing record specific retrievals”, on page 4-61.

Record complexes

Records from linked sets that share the same SI value form a *record complex*. Detail records from a set that is linked to an OMNIDEX master are indexed individually by their relative record numbers, and as part of a record complex by their OMNIDEX SI values. You can index detail records by their OMNIDEX SIs alone (as record complexes) by specifying the Record Complex (RC) key option during installation.

When searching Record Complex keys in a detail set, records that share a common SI value are treated as one logical entity (a record complex) during a keyword search. The qualifying count returned after a keyword search represents the number of record complexes (each represented by its unique SI value) that qualified, not the number of individual detail records (each represented by its relative record number). Therefore, searches on several Record Complex keys qualify detail chains whose records collectively satisfy the criteria. In other words, the keyword argument for one key field can exist in one detail record, while the keyword argument for another key field might exist in another detail record in the same record complex.

In the example on the next page, the master data set CUSTOMERS and the detail data sets ORDER-LINES and SHIP-TO are in the same OMNIDEX domain. That is, ORDER-LINES and SHIP-TO were linked to CUSTOMERS during installation. Therefore, they all share the same OMNIDEX SI (here, CUSTOMER-NO). On a record level, records that share the same SI value (here, 4356) comprise a record complex.

Suppose that PRODUCT and ORDER-DATE were installed as keyword keys with the Record Complex option. Searches on these keys would return record complexes (SIs), not individual detail records (relative

record numbers). For example, if you search the ORDER-LINES detail data set for records of customers who ordered magnetic tape (MAG@ AND TAPE) in November of 1988 (8811@), OMNIDEX qualifies one SI, CUSTOMER-NO 4356.

CUSTOMERS (MASTER)

CUSTOMER-NO = 4356
COMPANY = Hewlett-Packard
CITY = Cupertino
STATE = CA
ZIP = 92374
PHONE = 408 555-4444

ORDER-LINES (DETAIL)

CUSTOMER-NO = 4356
PRODUCT = Mag Tape
ORDER-DATE = 881015

CUSTOMER-NO = 4356
PRODUCT = RS232 Cable
ORDER-DATE = 881109

SHIP-TO (DETAIL)

CUSTOMER-NO = 4356
ITEM-SHIPPED = Toner Cartridge
ADDRESS = PO Box 23
CITY = Reno
STATE = NV
ZIP = 56943

CUSTOMER-NO = 4356
ITEM-SHIPPED = Mag Tape
ADDRESS = 1200 East Diehl Rd.
CITY = Naperville
STATE = IL
ZIP = 60566

Note that the records in this SI domain qualify even though the keywords are not all contained within the same record. "Mag Tape" occurs in one record of the ORDER-LINES detail set, while "881109" occurs in a different record. These records qualified because the keys searched were Record Complex keys, and the keyword values "Mag Tape" and "881109" were indexed by the SI value of the records that contained them.

After a search qualifies the records in this record complex, you can retrieve any of them: the CUSTOMERS master record, or any of the detail records from either the ORDER-LINES or the SHIP-TO chain.

When keyword keys in different data sets within an SI domain are grouped together during installation, retrievals may be complicated further. Suppose, for example, that the Record Complex ITEM-SHIPPED key in the SHIP-TO detail has been grouped with the CITY key in the CUSTOMERS master. A search for a toner cartridge order from Cupertino would retrieve the record complex with an SI value of 4356, even though "Toner Cartridge" and "Cupertino" are in different records.

In short, all of the records in an SI domain that share a common OMNIDEX SI value are treated as a unit during a keyword search. Therefore, when an SI value is qualified during a search, all of the records in its record complex can be retrieved. The qualifying count returned for a keyword search on an SI domain reflects the number of SIs, or record complexes, not the number of records. This means that although there are five records in this SI domain, the qualifying count for this search would be 1.

Split Retrievals

You can search across Record Complex keys and record specific keys. For example, you can search regular keyword keys in a master set, and then search record specific (non Record Complex) keys in a related detail set.

This is called a *split retrieval*. The results of a split retrieval depend on whether the record specific or Record Complex key is searched first, and on whether a leading AND or NOT Boolean operator is used in the second search.



An OR operation is not supported for a split retrieval.

If you first qualify records using a record specific key in a detail, and then further qualify them using a Record Complex key, whether the second search returns the number of individual detail records (record specific), or the number of Record Complexes depends on the leading Boolean operator used in the second search.

- An AND operation returns an internal ID list that reflects SIs and therefore record complexes.
- An AND NOT operation returns an internal ID list that reflects the previous search.

For example, if you search a record specific (non Record Complex) key, you may qualify 20 individual detail records. Note that these 20 records may occur in 10 detail chains, and therefore correspond to 10 OMNIDEX SIs (record complexes).

If this list of 20 records is refined in an AND operation against a keyword key in the master (record complex), the resulting internal ID list reflects record complexes, not the number of detail records.

However, an AND NOT operation maintains the original type of ID list. That is, if you search a Record Complex key, and refine the record list using an AND NOT operation against a record specific key, the resulting qualifying count represents record complexes.

If you search a record specific (RS) key after searching a Record Complex (RC) key, an AND operation returns a record specific qualifying count, because the record specific key was the last field searched. An AND NOT operation returns record complexes since the first search was done on a Record Complex key. The results are summarized in the table below.

Operation	AND or	NOT or -	OR or +
RC followed by RS	RS results	RC results	not supported
RS followed by RC	RC result	RS results	not supported

Table 4-8: Split Boolean retrievals on RS keys

Compressing record specific retrievals

If your search returns a record specific result, you can “compress” the internal ID list and qualifying count to reflect the number of record complexes (SIs). In this way, you can still retrieve a record complex when searching record specific keys. You can use list compression to support OR operations in split retrievals.

You also can also use list compression to find individual records in a detail, then find their corresponding OMNIDEX master records, and finally find individual records in a second detail in the same domain.

For example, you could qualify records in a CUSTOMER-NOTES detail using a record specific DATE key. After compressing the ID list to reflect record complexes, you could continue the search against the corresponding CUSTOMERS master set. For example, you could search the STATE key to see how many customers from California purchased on a particular day.

Finally, you could search a record specific key in another linked detail set. For example, you could search the PRODUCTS key of the related ORDER-LINES detail to see how many terminals were purchased on a given day by customers in California.

To compress a list of individual detail records (relative record numbers) to record complexes (SI values), call ODXFIND mode 30.

Detail Record (DR) domains

Detail sets that aren't linked to a master during installation are said to be Detail Record (DR) Indexed. DR indexed details are indexed only by their relative record numbers, not by their SI values. Any keyword retrieval on a DR (unlinked) detail set qualifies individual records, not record complexes. For example, a keyword search on the INV-DATE and INV-QTY keyword keys of the INVENTORY detail set would qualify only those detail records that satisfied the arguments entered for both keys.

A DR indexed set is treated as an OMNIDEX domain unto itself. This means that you cannot perform searches across a DR detail and a master set as you can when the detail set is linked to the master. For example PRODUCTS and INVENTORY have a TurboIMAGE path between them, but they were not linked during installation, you could not qualify PRODUCTS master records, and then use those SI values to directly qualify the related INVENTORY detail records. Similarly, you can't extend a keyword key search across unlinked detail sets, even if the sets have TurboIMAGE paths to a common master set.

When sets are not linked, and therefore not in the same domain, use Multifind to perform keyword key searches across them. By installing OMNIDEX on the search items of unlinked sets, or any item that creates a correspondence between unlinked sets, you can search across them. Multifind is described next in the "Multifind" section.

Multifind

Multifind allows keyword searches across unlinked data sets, or OMNIDEX domains. With Multifind, you can search on keyword keys across domains, and databases, as long as the search involves some common item that serves as a logical link between two sets. This can be:

- ❑ a common search item, as in the PRODUCTS master and the INVENTORY DR detail set.
- ❑ a keyword key that contains data values common to both sets, like the INV-DATE field of INVENTORY and the DELIVERY-DATE field of ORDER-LINES.

When tables share a search item (SI)

Data sets may share common SI values, but exist in different domains, as in the following cases.

- ❑ Both data sets may be master data sets, which precludes them from being linked and occupying the same OMNIDEX domain. An example might be a PROSPECT master, and a CUSTOMER master, where the unique identifier for a new CUSTOMER record is taken from the original PROSPECT record's SI. Similarly, you may have a current master set and a historical master set (possibly in a different database), with related information for a particular entity in both the current and history sets.
- ❑ Sets may have a master detail relationship, but exist in different domains. This can occur either when the detail data set is linked to a different master, or when the detail data set is not linked to any master.

When tables are not in the same OMNIDEX domain, you must use Multifind to continue a keyword search across them.

When tables share common data values

Two sets in different domains may share common data that is not in a TurboIMAGE SI. In an order entry system, for example, PO-NUMBER may be the SI in a PO-MASTER data set, but not in the ORDER-MASTER data set. Similarly, a social security number field (like SOC-SEC-NO) may exist in EMPLOYEE records for both a payroll database and a personnel database, but may not be a TurboIMAGE SI in either case.

Requirements

Multifind relies on keyword keys. It uses the data values of qualified records as keyword arguments against a linking item, which must be a keyword key in the target set. The target set can be part of another domain, or another database.

For Multifind to work, the following conditions must be met.

- ❑ The relational item in the target data set must be a keyword key.
- ❑ When the relational item in the target set contains embedded spaces or special characters, you should install the No Parse (NP) option on it. The No Parse (NP) option assures that data values in the target key are not parsed into smaller, meaningless keyword values by OMNIDEX parsing. See “No Parse”, on page 2-10, for more information about the No Parse option.
- ❑ There must be a meaningful correspondence between the data taken from the source set, and the data used to search the target key (relational item).

If, in our previous example involving a PROSPECT master set and a CUSTOMER master set, the unique identifier assigned to a new CUSTOMER record had no correlation to the old PROSPECT record, then no meaningful correspondence would exist, and a Multifind search would return meaningless results. This might occur when the values for an SI field are randomly assigned, as in the case of an OMNIDEX SI/ID.

Often, you can solve this problem by adding a field to the target set to serve as a relational item, and installing a keyword key on it. This maintains the correlation between records. If the CUSTOMERS master set's SI/ID (CUSTOMER-NO) is randomly assigned, you could add a PROSPECT-NO field to the set to support Multifind searches from the PROSPECTS master set.

Executing a Multifind operation

To perform a Multifind search, use an ampersand (&), in the keyword list of the ODXFIND intrinsic. You can also pass the name of a file that contains stored data values to be used in the search. The Multifind operator (&) or expression must be the only argument in a keyword list. Any keywords or operations that follow a Multifind operator or expression (& or *&filename*) are ignored.

In a Multifind search, ODXFIND or DBFIND searches the target key referenced in the *field* or *item* parameter, using data values from the previously qualified records as keyword arguments. The data values that are used in the retrieval can either reside in memory, or in an external file. Both means of supplying keyword arguments are discussed below.

Multifind from memory

If the argument values used against the target set's key are OMNIDEX SI values, then they can be supplied from memory. In this case, only the Multifind operator (&) is passed to the ODXFIND *keywords* parameter. This method provides the best Multifind performance.

You would use this method, for example, to qualify records in a detail that belongs to a domain other than the current domain. For example, to qualify records in the PRODUCTS master set, and then find their INVENTORY records, you would need Multifind (because INVENTORY is in a different OMNIDEX domain than PRODUCTS).

Given a hypothetical OMNIDEX application program, the search might proceed like this:

1. Search the key of the PRODUCTS data set using the keyword argument **TERMINAL**:

Data Set: **Products**

Product Name: **TERMINAL**

16 entries qualify, do you want to list (L), to print (P), or another data set (D) [RE-QUALIFY]: **D**

2. Access the INVENTORY detail set:

Data Set? **INVENTORY**

3. Next, determine how many terminals were on hand in June 1988. Begin by entering an ampersand (&) as an argument against the PRODUCT-NO keyword key.

```
Product-No: &
```

This enters the PRODUCT-NOs of the previously qualified records as keyword arguments combined in a Boolean OR operation. If the previous search on PRODUCTS qualified the PRODUCT-NOs 2623A, 2624B, 2392A-049, 2392A-160 and 2623B, entering & has the same effect as entering:

```
Product-No: 2623A+2624B+"2392A-049"+"2392A-160"+2623B
```

This qualifies the 35 INVENTORY records for the five types of terminals qualified in the previous search.

```
35 entries qualify, do you want to list (L), to  
print (P), or another data set (D) [RE-QUALIFY]:  
[return]
```

The qualifying count that results from this Multifind search is greater than the qualifying count from the search on PRODUCTS. This is because the search on INVENTORY reflects the number of individual detail records, not the number of record complexes.

4. Then, you could qualify the records further with respect to date:

```
Inventory-Date: 8806@
```

```
12 entries qualify, do you want to list (L), to  
print (P), or another data set (D) [RE-QUALIFY]: P
```

Multifind from an external file

When the relational item is not the OMNIDEX SI of the source data set, or when searching across databases, Multifind from memory is not supported. Instead, you can transfer some field's values for all qualified records to a file. Then, you can use that file (those values) as arguments against a keyword key in the target domain. This key, called the *target key*, should provide a logical link between both sets involved in the search.

There are two ways to create such a file using the ODXTRANSFER intrinsic. They are discussed next.

Writing OMNIDEX SIs to an external file

ODXTRANSFER modes 1 and 2 write OMNIDEX SIs to a file specified by the calling program. Take, for example, the search for “TERMINAL” originating in the PRODUCTS master set. After the search against PRODUCT-NAME, you could call ODXTRANSFER mode 1 to create a five record file (which you would name) to contain the five qualified PRODUCT-NOs for terminals.

You could pass this file as an argument (in *keywords*) against the PRODUCT-NO key (the target key) of INVENTORY. To use a file in a Multifind, follow the Multifind operator (*&filename*) with the name of the file. So if the file was created as PRODID, the Multifind argument against the target key would be &PRODID. As with the earlier example, all INVENTORY records containing one of the five PRODUCT-NO values would be qualified.

We could have used just the Multifind operator to pass the PRODUCT-NOs from memory. But if the INVENTORY data set was in a different database than PRODUCTS, you would have to transfer the PRODUCT-NOs to an intermediate file.

Writing data fields to an external file

When the target key that forms the logical link between sets is not the OMNIDEX SI of the source data set, you must use ODXTRANSFER mode 101 or 102 to write the contents of a selected field to an external file. See ODXTRANSFER in the *OMNIDEX ImagePlus SDK API Guide* for more information on ODXTRANSFER mode options.

A common example of this technique is when an initial search is done on an unlinked (DR) detail, and you wish to qualify related master records. Because OMNIDEX uses relative record numbers, not SIs, to identify DR detail records, you cannot do a Multifind from memory. You can transfer the TurboIMAGE SI values of the qualified detail records to a file, then use that file to do a Multifind search against the SI of the master (assuming it is a keyword key).

You could pass the data values in this file as arguments through an ODXFIND. You would reference the target key in the *field* parameter, then pass the file's name preceded by an ampersand in the *keywords* parameter (*&filename*), as in the example above.

Maintenance

Normally, OMNIDEX databases don't require maintenance, except in certain situations. This section discusses these situations and the maintenance they require, as summarized in the table below.

Conditions	Required procedures ^a
DBUNLOAD/DBLOAD	2, 3
Update sets with Batch Indexed keys	2, 3 ^b
Update sets using IMAGE-only mode	2, 3
RELOAD detail sets that have OMNIDEX keys	2, 3
Copy a database without using DBMGR ^c	1, 2, 3, 4
Rename a database without using DBMGR ^c	1, 2, 3, 4
Add or remove values from the excluded words list	3
Customize the translation of 8-bit characters	3
Change key options	2, 3
Recover from system failure	2, 3 ^d
Recover from passive indexing errors	2, 3
Structural changes ^e	1, 2, 3

Table 4-9: Conditions requiring OMNIDEX maintenance

- a. The required procedures are number-coded as follows:
 1. Reinstall OMNIDEX keys.
 2. Reindex sorted keys.
 3. Reindex keyword keys.
 4. Enable TPI (if previously enabled).
- b. If the operation only affects one type of key (only sorted, for example) you need only reindex that type of key.
- c. Vendor dependant software may support TPI without extraordinary measures. Consult your database tools vendor for complete information.
- d. Reindexing is not necessary if the database was enabled for TPI during the system failure.
- e. Structural changes constitute any change to a database (like adding a data set) that affects the numbering of TurboIMAGE data sets and items.

Changes may be made with a variety of database management utilities, including Hewlett-Packard's DBChange or Adager. However, DBMGR, DISC's database management utility, is designed to maintain the OMNIDEX indexes when you rename or copy a database.

The procedures that do not require any reinstallation or reindexing are:

- change a set capacity
- change a primary path (if details are not reloaded afterwards)
- store or restore a database

The conditions and procedures described above are discussed next.

Reinstalling OMNIDEX keys

People often reinstall OMNIDEX on a database to change the keys or key options that are already in place. There are other situations that require you to reinstall OMNIDEX, for example:

- when you add or delete TurboIMAGE fields or sets
- when the OMNIDEX root file is damaged or corrupted

You should generate an installation job after your initial installation. You can then stream the job to reinstall in situations like the ones mentioned above. You can use OmniUtil to generate an installation job by selecting `3. Generate Installation Job` from the `Index Installation` menu.

This section first discusses modifying how OMNIDEX keys are installed on your database. Next, it addresses some of the repairs or changes to a database that require reinstallation. Finally, it discusses installation jobs.



Reinstallation requires Privileged Mode in the DISC account, and exclusive access to the index files built by OmniUtil.

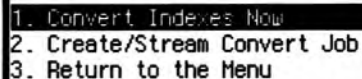
Updating an existing OMNIDEX installation

If you are upgrading from a previous version of OMNIDEX (2.02 or later) you can use OmniUtil to upgrade your databases automatically. Check your *OMNIDEX Software Bulletin* to see if any additional conversion measures are required for your previous version of OMNIDEX.

1. First, log on to the database's group and account.
2. Run Omniutil as discussed on page 3-2.
3. From the OmniUtil Main Menu (shown in Figure 3-1 on page 3-6), select 1. Index Installation and Maintenance.
4. From the Index Installation submenu, select 6. Convert Previous Version's Indexes. If you are upgrading from a previous version of OMNIDEX, this option lets you install current indexes that provide the same access (keyword or sorted, plus any key options) that you had under your previous version.

OmniUtil prompts you for a database. Either enter the name of a database, or press [f2] and select a database from the pick list that OmniUtil displays.

5. When you select a database, OmniUtil asks:



```
1. Convert Indexes Now
2. Create/Stream Convert Job
3. Return to the Menu
```

If you select 3. Return to the Menu, you are returned to the Index Installation and Maintenance menu.

If you select 2. Create/Stream Convert Job, OmniUtil prompts you for the passwords necessary to stream a job, and offers you the options (OK) and (Cancel). If you select (OK), OmniUtil asks if you want to save or stream the index conversion job file.

If you choose 1. Stream the job file, OmniUtil prompts you for MPE STREAM command options, then streams the job, closes the database, and returns you to the Index Installation and Maintenance menu. If you choose 2. Save the job file, OmniUtil saves the file as the database's name, plus the letters "CV" in the group where you're logged on. If a file with that name already exists, OmniUtil offers you the options of overwriting the existing file, or saving the file with a different name.

6. If you select 1. Convert Indexes Now, OmniUtil asks, Do you want to convert existing indexes? If you select (No), you are returned to the Index Installation menu. If you select (Yes), OmniUtil tells you that it is reading your old installation.
7. After reading your current installation, OmniUtil updates it to an identical current installation, and tells you when the update is finished. When you press (OK) OmniUtil displays this menu:

```

1. Populate All Indexes Now
2. Create/Stream Indexing Job
3. Set ID Sort Order
4. Set Index Buffer Size
5. Return to the Menu

```

- | | |
|-------------------------------|--|
| 1. Populate All Indexes Now | loads the index structures immediately and online. |
| 2. Create/Stream Indexing Job | creates a job file that loads the index structures, and lets you schedule a time for the job to stream. |
| 3. Set ID Sort Order | lets you index records in an OMNIDEX domain in sorted order by specified fields. When you select this item from the menu, OmniUtil only displays those sets that will support the sorting of OMNIDEX IDs. |
| 4. Set Index Buffer Size | lets you specify how much memory (in megabytes) to dedicate to the indexing operation. You must enter a value between 2 and 128, as discussed in "Other Performance Considerations", on page 4-96. Menu option 5. Return to the Menu returns you to the Reindex All Indexes submenu. |
| 5. Return to the Menu | takes you to a dialog box that asks, Skip Indexing Operation? If you select (Yes), OmniUtil returns you to the Index Installation and Maintenance menu without loading the new indexes. |

After you update the installation, you can change it to install new keys or options, as described in chapter 2 "OMNIDEX Installation".

Changing key installation

You can add or remove OMNIDEX keys, or change their key options by reinstalling OMNIDEX. To change the way OMNIDEX is installed on a database, run OmniUtil as discussed on page 3-2.

Select 1. Index Installation and Maintenance from the main menu. Then, select 1. Install or Modify Indexes from the Index Installation menu. Follow the procedures outlined in “Changing the Installation”, on page 2-34.

Modifying keys interactively

To modify keys interactively, run OmniUtil and select 1. Install or Modify Indexes from the Index Installation menu. The prompts for modifying an installation are the same as a first time installation.

Indexing keys in batch

After changing keys on a large database, you may want to use a job stream to reindex the changes. After activating the installation, select 2. Create/Stream Indexing Job.

Reinstalling as a maintenance measure

The following situations require you to reinstall OMNIDEX. Some of them may also require you to delete or reload some or all of the indexes, as discussed earlier.

Fixing a corrupt OMNIDEX root file

The OMNIDEX root file (named *dbnameOA*) contains information about how OMNIDEX keys are installed on a database. It contains information about OMNIDEX data sets, keys, and options, based on the TurboIMAGE schema and TurboIMAGE root file at the time OMNIDEX was installed.

If the OMNIDEX root file is damaged, you must reinstall OMNIDEX. Reindexing may also be required if updates have been made to the OMNIDEX database while the OMNIDEX root file was corrupt.

When TurboIMAGE changes affect OMNIDEX

If you change the structure of your TurboIMAGE database, copy it, or rename it, you may need to reinstall OMNIDEX.³ The situations that warrant this are:

- ❑ When you copy a database using a utility other than DBMGR, the OMNIDEX indexes are not automatically copied. Therefore, you must reinstall OMNIDEX.
- ❑ When you rename a database using a utility other than DBMGR, the OMNIDEX indexes still reflect the old database name. Therefore, you must reinstall OMNIDEX.
- ❑ When you insert an item to or delete an item from the middle of the item definition list, you must reinstall OMNIDEX.

If items are appended to, or deleted from, the end of the item definition list, no maintenance is necessary.

- ❑ When you add items to, or delete items from a data set that contains OMNIDEX keys, you must reinstall OMNIDEX.
- ❑ When you define a new data set, or delete an existing set, that occurs before another set in the set definition portion of the schema, you must reinstall OMNIDEX.

In general, if you add items or sets that change the TurboIMAGE item and set numbers of keyed fields, fields used in composite keys, or sets that contain OMNIDEX keys, you must reinstall OMNIDEX and reindex all keys.

Copying or renaming a database

If you copy or rename a database using any utility other than DBMGR, you must reinstall OMNIDEX on the new copy. Because DBMGR is designed for use with OMNIDEX databases, it automatically copies or renames the OMNIDEX index files when it copies or renames a database.

3. Your database tool may support OMNIDEX indexes with no additional measures. Consult your database tool vendor for complete information.

To copy a database and its OMNIDEX indexes, log onto the account where you want the database to reside. Run DBMGR by entering the following run statement at a system prompt:

```
RUN DBMGR.PUB.DISC;LIB=G
```

1. Enter the name of the database you wish to copy, then enter its maintenance word if necessary:

```
Database name: SALES.DEMODB.DISC
```

```
Maintenance word:
```

2. Enter COPY, or COP, at the DBMGR: command prompt:

```
Enter a command at the prompt or ? for help.
```

```
DBMGR: COP
```

```
Copy database: SALES.DEMODB.DISC
```

3. Enter the name you want to assign to the new copy. You can qualify the name to the group level, as in the example below.

```
Name to be assigned to the new copy ? SALES.TESTDB
```

DBMGR lists the data sets and OMNIDEX index files as it copies them, then informs you when the copy operation is complete. Note that if the original database is enabled for TPI, the copy is enabled for TPI as well.

To rename a database, log onto the group and account where the database resides. Run DBMGR as shown above. Enter the name of the database you wish to rename, then enter its maintenance word if necessary.

1. Enter **RENAME**, or **REN**, at the **DBMGR :** command prompt:

```
DBMGR: REN
```

2. Enter **BASE** at the next prompt:

```
Rename the database, a data set, or a data item  
(Base/Set/Item/[Exit]): BASE
```

3. Finally, enter the new name for the database:

```
New name ? [no change] TIM
```

DBMGR renames the database and OMNIDEX index files, and tells you when it is finished.

```
Working  
Renaming OMNIDEX Index files  
Rename complete. New name: TIM.TESTDB.MYACCT
```

4. Enter **E** to exit DBMGR, and press **[return]** at the Database name: prompt.

Installation jobs

You can generate a job for reinstallation using OmniUtil, or write one yourself using a Hewlett-Packard Edit/3000 compatible text editor.

Considerations for reinstalling in batch

Installation in batch does not require activation. An installation job stream automatically creates the necessary data sets and items. Therefore, you should only stream or schedule the job for a time when you have exclusive access to the OMNIDEX indexes.

A sample installation job file, shipped on your software tape as SALESIN.DEMODB.DISC, appears below. The job is listed on the next few pages

```

!JOB SALESIN, MGR. DATA, DEMODB; PRI=DS; OUTCLASS=LP, 1, 1
!COMMENT
!COMMENT *****
!COMMENT Created: WED, JUN 17, 1992, 3:36 PM
!COMMENT
!COMMENT SALESIN is an installation JOB stream generated
!COMMENT for an existing OMNIDEX-enhanced database. Use
!COMMENT it to reinstall OMNIDEX on the database, as an
!COMMENT archival copy of the JOB, or just to see how
!COMMENT OMNIDEX is installed on your database.
!COMMENT *****
!COMMENT
!SETJCW CIERROR = 0
!CONTINUE
!RUN OMNIUTIL.PUB.DISC
INSTALL SALES.DEMODB
;
TABLE CUSTOMERS
CREATE CUSTOMER-NAME MULTIPLE SORTED OPTIONS BI G1
CREATE CONTACT MULTIPLE OPTIONS BI G1
CREATE TITLE MULTIPLE OPTIONS BI G1
CREATE CITY MULTIPLE OPTIONS G2
CREATE STATE MULTIPLE OPTIONS G2
CREATE ZIP MULTIPLE OPTIONS G2
CREATE SALESMAN MULTIPLE
CREATE CUSTOMER-BAL MULTIPLE OPTIONS NP
CREATE COMMENTS MULTIPLE
CREATE LAST-PUR-DATE SORTED
CREATE S-D SORTED FROM STATE LAST-PUR-DATE [1:4]
;

```

```
TABLE ORDER-LINES LINK TO CUSTOMERS
CREATE PRODUCT-NO MULTIPLE
CREATE DELIVERY-DATE MULTIPLE OPTIONS RC
CREATE PROD-DD SORTED FROM PRODUCT-NO DELIVERY-DATE
;
TABLE CUSTOMER-NOTES LINK TO CUSTOMERS
CREATE DATE-ENTERED MULTIPLE SORTED
CREATE ACTION MULTIPLE
CREATE CUST-DATE SORTED FROM CUSTOMER-NO DATE-ENTERED
;
TABLE PRODUCTS TRANSPARENT
CREATE PRODUCT-CLASS MULTIPLE
CREATE PRODUCT-NAME MULTIPLE OPTIONS G1
CREATE PRODUCT-DESC MULTIPLE OPTIONS G1
;
TABLE ORDER-MASTER
CREATE CUSTOMER-PO MULTIPLE OPTIONS NP
CREATE SALES-DATE MULTIPLE
;
TABLE INVENTORY
CREATE PRODUCT-NO MULTIPLE OPTIONS NP
CREATE INV-DATE MULTIPLE
CREATE INV-LOC MULTIPLE
;
ACTIVATE
EXIT
!IF CIERROR <> 0 THEN
! SHOWJCW
! TELL MGR.DATA; SALESIN: Installation Failed !!!
!ELSE
! TELL MGR.DATA; SALESIN: Installation completed.
!ENDIF

!IF CIERROR > 1609 AND CIERROR < 1628 THEN
! COMMENT - TELL Message failed
! SHOWJCW
! SETJCW CIERROR=0
!ENDIF
!EOJ
```

Reindexing

Reindexing refers to erasing and reloading the OMNIDEX indexes. During normal updating of TurboIMAGE data, the OMNIDEX indexes can also be updated synchronously (see page 4-50). Sometimes, however, it is beneficial, or necessary, to update the indexes by reindexing.

To reindex an OMNIDEX database, table, or key, run OmniUtil. Select 2. Reindexing Options from the Main Menu, and select the indexing option you want. The situations that require reindexing are discussed below.

When is it necessary?

Reindexing is necessary when the data in the indexes does not accurately reflect data in the database. You must reindex when:

- you add or modify keys or their options.
- you reload detail sets.
- you perform IMAGE-only updates, or update Batch Indexed keys.
- you change the excluded words list (affects only keyword keys).
- you change how 8-bit characters are translated.
- you incur Passive Indexing Errors.

Changes to installation

When you change how keys are installed, OmniUtil asks you if you want to reindex those changes. You should reindex as soon as possible after activating a changed installation. You can reindex either in an OmniUtil session, or by streaming a job.

Set reload operations

After reloading a detail set, you must reindex all sorted keys installed on the set, and reindex the domain in which the set is installed. This is because reload operations affect the relative record numbers that OMNIDEX uses to get detail records. After a reload on a set, a detail record may have moved to a different record address, which may not be the address reflected for it in the OMNIDEX indexes.

After a DBUNLOAD and DBLOAD, you must reinstall OMNIDEX and reindex the entire database.

Re-specifying excluded words

After you re-specify the excluded words, you must reindex all the OMNIDEX keys. If you do not, the keywords stored in the OMNIDEX indexes do not reflect the most recently specified excluded words. This means that currently excluded keyword values may be present in the indexes, while previously excluded, but currently valid, keyword values may not be present.

Customizing the Translation of 8-bit characters

If you add or change a translation table for any database to customize how 8-bit characters are indexed, you must reindex the keyword keys for that database. If you do not reindex, the keywords in the indexes for that database won't reflect the custom translation you configured in your translation table. See "Customizing the Translation of 8-bit Characters", on page 2-24, for more information about translating 8-bit characters.

Recovering from Passive Indexing Errors

If you enabled Passive Error Handling for an XL, update applications that reference that XL do not terminate when indexing errors occur. Instead, indexing errors are logged to a file created in the group where the database resides. The file is named for the database, plus the letters "LG". Whenever you try to open an OMNIDEX database where indexing errors occurred, DBIOPEN or DBOPEN issues a warning:

```
OMNIDEX communication from program: Previous indexing
errors found for database dbname in file file.group.account.
```

This means that the indexes do not accurately reflect the data contained in the database. To restore the indexes and purge the error log file, reindex the database using OmniUtil.

For more information about Passive Error Handling, see the "Error handling" section in the "Programming" chapter of your *OMNIDEX ImagePlus SDK API Guide*.

IMAGE-only updates

If you add or delete records using an IMAGE-only update mode, and TPI is not enabled for the database, you must reindex the updated data sets. IMAGE-only updates do not update the indexes that correspond to OMNIDEX keys. Therefore, sorted and keyword retrievals are not accurate after an IMAGE-only update.

Batch indexing option

If you have the Batch Indexing (BI) option installed on any keys, you must periodically reindex them. As the name suggests, the indexes associated with Batch Indexed keys are only updated during a reindexing operation.

If the Batch Indexing option is installed on a sorted key, you must periodically reindex that key. If the Batch Indexing option is installed on a keyword key, you must periodically reindex that domain.

Reindexing jobs

Whether you choose to reindex in a job stream or interactively, depends on the number and size of keys you are reindexing, and how often you must reindex.

Reindexing jobs offer the following benefits.

- ❑ They can be scheduled to reindex at times of low system activity and exclusive access to a database.
- ❑ They can restore account capabilities, like Privileged mode (PM), only for the duration of the reindexing process.
- ❑ You can reindex in a batch priority queue and avoid unnecessary strain on system resources.
- ❑ They require little time and effort.

Maintenance and Third Party Indexing (TPI)

When Third Party Indexing (TPI) is enabled for a database, TurboIMAGE/iX intrinsics update its OMNIDEX indexes along with its TurboIMAGE data in real time. For such maintenance operations as changing data set capacities, real time updates to indexes can cause a performance bottleneck. Therefore, depending on the database management tool you use, you may need to disable a database for TPI before you change a data set's capacity. Some utilities, like ADAGER and DBMGR, are TPI aware and will not pose a performance issue.

Determining whether TPI Is enabled

There are several ways to tell whether or not a database is enabled for TPI. They are discussed next.

DBINSTAL

You can use DBINSTAL, which resides in PUB.DISC to see if a database is enabled for TPI. Just follow these steps:

1. Log on as the database creator.
2. Run DBINSTAL out of PUB.DISC:
`RUN DBINSTAL.PUB.DISC`
3. After it displays a program banner, DBINSTAL asks for the name of a database. Enter the name of the database you want to check for TPI:

```
Data base name: DIRIMG;Mode=44
TurboIMAGE TPI enabled
```

```
Cmd:
```

If the database is enabled for TPI, DBINSTAL tells you so. If no message appears, the database is not enabled for TPI.

You can also use DBINSTAL to enable or disable a database for TPI, as discussed later.

-
4. If you do not specify a shared access mode, DBINSTAL tries to open the database with exclusive access.

DBUTIL

You can use Hewlett-Packard's DBUTIL database utility to see if a database is enabled for TPI. Use the SHOW FLAGS command by following these steps:

1. Log onto the account where the database resides. If you are not the database creator, you must supply its maintenance word.
2. Run DBUTIL out of PUB.SYS:

```
RUN DBUTIL.PUB.SYS
```

3. At the DBUTIL command prompt (>>) enter SHOW *dbname* FLAGS, where *dbname* represents the name of the database that you are checking. For example, to check a database named SALES, you would enter:

```
>>SHOW SALES FLAGS
  For database DIRIMG

  Access is enabled.
  Autodefer is disabled.
  .
  .
  .
  Indexing is enabled for OMNIDEX.
  .
  .
  Subsystem access is READ/WRITE.
```

DBUTIL lists whether or not the database is enabled for TPI, among other things, and names the indexing product.

DBINFO mode 803

You can programmatically check to see if a database is enabled for TPI by calling DBINFO mode 803. See the "DBINFO" listing in your *OMNIDEX ImagePlus SDK API Guide*.

Turning TPI on or off

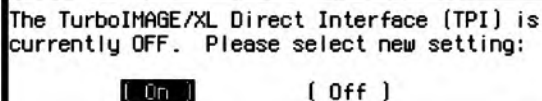
The recommended method for turning TPI on or off is to use either OmniUtil or DBINSTAL. Both methods are discussed next.

Using OmniUtil

OmniUtil is menu driven and simple to use. Because it is interactive, it is best suited to enabling or disabling one database for TPI.

To enable or disable a database for Third Party Indexing, be sure you have exclusive access to the database, and follow these steps:

1. Run OmniUtil as discussed on page 3-2.
2. From the Main Menu, select 5. Configuration Options, and then select 2. Database Options.
3. From the Database Options menu, select 1. Configure TurboIMAGE/XL Direct Interface. When you select this item, OmniUtil displays the following message:



```
The TurboIMAGE/XL Direct Interface (TPI) is
currently OFF. Please select new setting:
( On )          ( Off )
```

If you select (On) the indexes are recognized by the Standard Interface to Third Party Indexing intrinsics, and are automatically updated in real time as TurboIMAGE data is updated. If you select (Off) the indexes are not updated as TurboIMAGE data is updated. Note that this feature is available only with TurboIMAGE version C.04.03 or later, or MPE/iX version 4.0 or later.

Using DBINSTAL

DBINSTAL is DISC's command driven database utility. Because it accepts commands, it is callable from job streams. As such, it is well suited to enabling and disabling TPI for multiple databases.

Use DBINSTAL's SET TPI command to turn TPI on or off for a database. To use DBINSTAL interactively:

1. Log on as the database creator with exclusive access to the database.

2. Run DBINSTAL out of PUB.DISC:

```
RUN DBINSTAL.PUB.DISC
```
3. After it displays a program banner, DBINSTAL asks for the name of a database. Enter the name of the database you want to check for TPI:

```
Data base name: DIRIMG
TurboIMAGE TPI enabled

Cmd:
```
4. Then, at the `Cmd:` prompt, enter `SET TPI ON` to enable TPI, or `SET TPI OFF` to disable TPI. Enter `QUIT`, to exit DBINSTAL.

If you are enabling or disabling TPI for several databases, as when upgrading from a previous version of OMNIDEX, you can create and stream a job to do this. Below is a job named "TPION" that enables TPI for two databases, SALES and ACCTS:

```
!JOB TPION,MGR.DATA,PUB;PRI=DS;OUTCLASS=LP,1,1
!SETJCW CIERROR = 0
!CONTINUE
!RUN DBINSTAL.PUB.DISC                                (Runs DBINSTAL)
SALES                                                  (Specifies the first database)
SET TPI ON                                           (Enables TPI)
EXIT                                                 (Exits to the next database)
ACCTS                                               (Specifies the next database)
SET TPI ON
QUIT                                                (Exits to the operating system)
!IF CIERROR <> 0 THEN
! SHOWJCW
! TELL MGR.DATA; TPION: Job stream Failed !!!
!ELSE
! TELL MGR.DATA; TPION: Job stream completed.
!ENDIF
!IF CIERROR > 1609 AND CIERROR < 1628 THEN
! COMMENT - TELL Message failed
! SHOWJCW
! SETJCW CIERROR=0
!ENDIF
!EOJ
```

You could use a similar job stream to disable TPI by changing "SET TPI ON" to "SET TPI OFF" for all occurrences.

Index updates and the Transaction Manager

The Transaction Manager logs TurboIMAGE update transactions so that you can recover data if a catastrophe, such as a system failure, occurs in the middle of an update transaction. When TPI is enabled for a database, the Transaction Manager logs updates to OMNIDEX indexes as well as updates to TurboIMAGE data. Although logging index update transactions is beneficial in the event of a system failure, it has some drawbacks.

- ❑ The amount of information stored in the log file increases with the volume of index update transactions. As a result, a high volume of index updates can fill the log file in the course of a transaction. This causes a process abort with the error `Stalled Transaction: Exceeded Accumulated Log Data Threshold`.
- ❑ Logging each index transaction requires additional I/O. Depending on the size of the domain where the indexed set is installed, and the number and types of OMNIDEX keys installed on the set, update performance can suffer.

Either of the drawbacks mentioned above are compounded when any of the following conditions exist:

- ❑ Updating a large number of keywords increases the amount of information written to the log file.
- ❑ Rearranging IDs affects the relative positions of many indexed values, and therefore causes much information to be written to the log file.
- ❑ Block splits cause the greatest impact to an index, and therefore cause the most amount of information to be written to the log file.

When performing a high volume of transactions that require updates to OMNIDEX indexes, you may want to disable the transaction manager as discussed next.



The information listed above only applies to updates performed using TurboIMAGE intrinsics against a TPI-enabled database.

When you exempt OMNIDEX indexes from logging and recovery by the transaction manager, you run a slight risk of losing index update transactions in the event of a catastrophe, such as a system failure. Recovering indexes from a catastrophe requires you to first recover the TurboIMAGE data that was lost, and then perform a reindexing operation of the keyword domains or sorted keys that you were updating when the catastrophe occurred. For large transactions, a reindexing operation typically provides better performance than logging all update transactions and recovering the indexes from the log file. For smaller transactions, it may be better to log index transactions

Exempting indexes from the Transaction Manager

To exempt OMNIDEX indexes from the Transaction manager, use DBINSTAL's SET TPI NOXM command to enable TPI while exempting the OMNIDEX indexes from transaction logging. Follow the steps below.

To use DBINSTAL interactively:

1. Log on as the database creator with exclusive access to the database.
2. Run DBINSTAL out of PUB.DISC.
`RUN DBINSTAL.PUB.DISC`
3. After it displays a program banner DBINSTAL asks for the name of a database. Enter the name of the database you want to check for TPI.
`Data base name: DIRIMG`
`TurboIMAGE TPI enabled`
`Cmd:`
4. Then, at the `Cmd:` prompt, enter `SET TPI NOXM` to enable TPI while exempting the OMNIDEX indexes from transaction logging, or `SET TPI ON` to enable TPI and transaction logging for OMNIDEX indexes. Enter `QUIT`, to exit DBINSTAL.

If you are enabling or disabling TPI for several databases, as when upgrading from a previous version of OMNIDEX, you can create and stream a job to do this. Below is a job named "TPINOXM" that enables TPI for two databases, SALES and ACCTS:


```

!JOB TPINOXM,MGR.DATA,PUB;PRI=DS;OUTCLASS=LP,1,1
!SETJCW CIERROR = 0
!CONTINUE
!RUN DBINSTAL.PUB.DISC                                (Runs DBINSTAL)
SALES                                                  (Specifies the first database)
SET TPI NOXM                                          (Enables TPI)
EXIT                                                  (Exits to the next database)
ACCTS                                                (Specifies the next database)
SET TPI NOXM
QUIT                                                  (Exits to the operating system)
!IF CIERROR <> 0 THEN
!  SHOWJCW
!  TELL MGR.DATA; TPINOXM: Job stream Failed !!!
!ELSE
!  TELL MGR.DATA; TPINOXM: Job stream completed.
!ENDIF
!IF CIERROR > 1609 AND CIERROR < 1628 THEN
!  COMMENT - TELL Message failed
!  SHOWJCW
!  SETJCW CIERROR=0
!ENDIF
!EOJ

```

You could use a similar job stream to enable TPI and transaction logging of OMNIDEX indexes by changing “SET TPI NOXM” to “SET TPI ON” for all occurrences.

Recovery from a system failure

How you recover from a system failure when updating records depends on whether TPI is enabled, and whether you are logging index transactions.

If you enabled TPI for a database using either DBINSTAL or OmniUtil, the database and its OMNIDEX indexes are attached to the Transaction Manager. Recovery of the OMNIDEX indexes is handled automatically when you recover your TurboIMAGE data.

If TPI is not enabled for a database, or if you used DBINSTAL to exempt OMNIDEX indexes from transaction logging (SET TPI NOXM), you must recover the OMNIDEX indexes by reindexing the affected keyword domains and sorted keys after you recover your TurboIMAGE data. Reindexing is discussed on page 4-78.

Maintaining access to OMNIDEX indexes

Sometimes, users are denied access to a released OMNIDEX database. This can be caused by the level of security specified for the group where the database, and its index files, reside.

OMNIDEX indexes are privileged MPE files. If the database to which they belong is enabled for TPI, they are automatically released when you release the database.

If TPI is not enabled for a database, index files are not automatically released when you release a database to which they belong. They are subject to the same access restrictions you place on any privileged file in that group. Therefore, OMNIDEX users must have read, write and lock access to any indexes.

Note that you can determine a user's access to an index file by logging on as the user in question. By entering `LISTF dbname0A.group.account,4`, where *dbname* is the name of the database causing access problems, you can determine the access for this user to the indexes in question.

To automatically release index files for a TPI-disabled database:

1. Enable the database for TPI (see page 4-81).
2. Release the database using DBUTIL.
3. Disable the database for TPI (see page 4-81).

Maintaining OMNIDEX index file capacities

Because OMNIDEX indexes are privileged MPE files, you can not maintain their capacities using a database utility like DBMGR. Under most conditions, their size is maintained automatically during OmniUtil indexing operations. If you are adding a high volume of data to a database – as when converting a prototype database to production – and you are indexing those records in real time, you should manually increase the size of the index files to prevent them from filling up.



Do not change the capacity of the OMNIDEX control file, which ends in the suffix "0A". Doing so could jeopardize your OMNIDEX installation.

How OMNIDEX determines index file capacities

OMNIDEX determines index file capacities differently for keyword and sorted keys.

For sorted keys, OMNIDEX calculates the appropriate index size based on the capacity of the data set where the key is installed. When calculating index file capacity for a sorted key, OMNIDEX allows for about ten times more data than the current data set capacity.

For keyword keys, OMNIDEX creates the files with a default size. During the first indexing after installation, and for any subsequent reindexing operations, OMNIDEX resizes the files based on the number of keywords unloaded. Again, OMNIDEX allows for growth of about a factor of ten.

Checking index file capacities

The index files for any database reside in the databases group and account. The index files are named for the database, and have a suffix that is sequentially assigned from "0A" to "9Z". The index file ending in "0A" is the OMNIDEX Control file that holds information about the other index files.

To check the index file capacities for a database, use CAPCHK, a command file that resides in the UTIL.DISC account.

When specifying a database in CAPCHK, you can use wildcards in the database name. You can also specify the database's group and account.

You can specify the database in the CAPCHK command line, as shown below:

```
CAPCHK SALES
```

If you enter CAPCHK by itself, without a database name, it prompts you for one.

```
:CAPCHK
```

```
This command file shows the percent full of index files  
for the specified database.
```

```
usage is :CAPCHK basename
```

```
Enter basename : SALES.DEMODB.DISC
```

If the database you specified does not exist, CAPCHK returns the following error:

```
LISTF failed error -431
```

When you correctly specify a database, CAPCHK lists all of the index files, except the OMNIDEX Control File (*dbnameOA*), along with their length, their limit and the percent that they are full.

The abridged example below shows capacity information for indexes associated with the sales database.

```
FILE NAME                               EOF / LIMIT = %FULL
SALES0B.DEMODB.DISC                     48 / 4118 = 1%
SALES0C.DEMODB.DISC                     104 / 816 = 12%
.
.
SALES0J.DEMODB.DISC                      9 / 4096 = 0%
SALES0K.DEMODB.DISC                     16 / 4096 = 0%
SALES0L.DEMODB.DISC                     24 / 4096 = 0%
```

You can use the information that CAPCHK returns to determine how full the indexes are for any given database. If they are too full, you can increase their capacities using the CAP command file, as discussed in “Changing the index file capacities”, on page 4-91.

Situations requiring increased index file sizes

Problems relating to index file sizes generally occur when adding a large amount of data to a database after installing OMNIDEX. This may be because OMNIDEX was installed on a small or empty test database and data set sizes were then increased without reindexing. Reindexing automatically changes index files to the proper capacities.

When index files are full, OMNIDEX returns error 152 during a call to DBIPUT or DBPUT.

Changing the index file capacities

You increase index file capacities using the command file CAP, located in the UTIL.DISC account. Capacities are specified as a percentage of the number of records divided by the file limit for each index.



NOTE

To use CAP on a database in a specified account or directory, you must be able to create a file in that account or directory. Otherwise, CAP will fail when it resizes an eligible index file.

The syntax for the CAP command is:

```
CAP basename cutoff newpercent
```

where:

- | | |
|-------------------|---|
| <i>basename</i> | represents the name of the database that uses the index files you want to resize. The database name can contain any wildcards supported by MPE (@ # ?). It can also include the group and account where the database resides, but no wildcards in the account name. |
| <i>cutoff</i> | represents the percent of capacity that determines whether an index file's capacity is increased. For example, if you specify a <i>cutoff</i> value of 80, CAP will increase the capacities of all index files that are 80% full or greater than 80% full. |
| <i>newpercent</i> | represents the percent of capacity after the index file is resized. For example, if you specify a <i>newpercent</i> value of 50, any index files beyond the <i>cutoff</i> value are resized to be only 50% full. |

The example below would find the index files of all databases in DEMODB.DISC and increase the sizes of those files that were at least 80% full so that they were only 50% full:

```
:CAP @.DEMOB.DISC 80 50
```

If you enter the CAP command alone, it prompts you for the database name, cutoff percentage and new percentage. For example:

```
:CAP
```

```
This command file prompts for a database name, a
"cutoff" percentage full and a new percentage.
It checks the index files for the database
and if they are "cutoff" percent full, it
changes their capacities to be "new" percent full.
```

```
usage is :CAP basename cutoff newpercent
```

```
Enter basename : @.DEMODB.DISC
```

```
Enter the cutoff percentage : 80
```

```
Enter the new percentage for these indexes : 50
```

Before changing set capacities, CAP tells you what it is going to do and asks for permission to proceed:

```
Changing SALES indexes greater than or equal to 80%
full to 50%.
```

```
Is this correct? Y
```

If you answer yes, CAP displays all off the index files associated with the specified database(s) and tells you whether it is changing index capacity.

```
FILE NAME                EOF / LIMIT = %FULL
SALES0B.DEMODB.DISC      48 / 4118 = 1%
SALES0C.DEMODB.DISC      104 / 130 = 80%
Changing capacity of SALES0C.DEMODB.DISC to 208 = 50%
Capacity successfully changed
.
.
SALES0K.DEMODB.DISC      16 / 4096 = 0%
SALES0L.DEMODB.DISC      24 / 4096 = 0%
```

Because anyone can copy or use this file, you may want to restrict access to CAP.

Deinstalling OMNIDEX keys



Before you remove the OMNIDEX keys from a database, be sure that you've created an installation job so that you can reinstall them later. Without an installation job, there is no way of knowing what keys were installed before you removed them.

You can remove OMNIDEX with either OmniUtil or DBINSTAL. OmniUtil is best suited to removing OMNIDEX interactively from a single database. DBINSTAL is best suited for removing OMNIDEX in a job stream from several databases. Both methods of removing OMNIDEX are discussed next.

Unless you are using a utility is TPI-aware (like DBMGR and latest version of Adager), you should deinstall OMNIDEX when changing the structure of an OMNIDEX database. Even when using a TPI-aware utility, you may want to remove OMNIDEX and reinstall after the structural change to avoid any performance issues. You may have to reindex the whole database anyway if you insert a set or item in the middle of the set or item definition list. If you deinstall OMNIDEX before you change structure, you have greater control over the process..

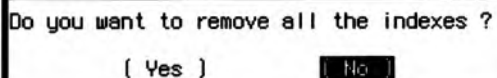
4

Removing OMNIDEX with OmniUtil

To remove OMNIDEX keys from a database, follow these steps:

1. Log on to the database's account.
2. Run OmniUtil as discussed on page 3-2.
3. From the Main Menu, select 1. Index Installation and Maintenance.
4. From the Index Installation menu, select 5. Remove All Indexes.
5. If you have not opened a database, OmniUtil asks you for one. Enter the name of a database, or press **[F2]** to select from one in your log on account.

6. OmniUtil asks you:



Do you want to remove all the indexes ?
[Yes] [No]

Select (Yes) to proceed.

7. OmniUtil tells you Please wait, removing all indexes. When it is finished removing the indexes, it tells you All indexes have been removed. Select (OK) to go on to the next operation, or to exit OmniUtil.

Using DBINSTAL

DBINSTAL is DISC's command driven database utility. Because it accepts commands, it is callable from job streams. As such, it is well suited to removing OMNIDEX from multiple databases.

Use DBINSTAL's Z! command to remove OMNIDEX from a database. To use DBINSTAL interactively follow steps 2 and 3 on page 4-81. Then, at the Cmd: prompt, enter Z! to purge the OMNIDEX root file and all the indexes. DBINSTAL asks PURGE root and all index files?. Enter Y to proceed. Enter QUIT, to exit DBINSTAL.

If you are removing OMNIDEX from several databases you can create and stream a job to do this. Below is a job named ODXOUT, that removes OMNIDEX from two databases:

```
!JOB ODXOUT, MGR.DATA, PUB; PRI=DS; OUTCLASS=LP, 1, 1
!SETJCW CIERROR = 0
!CONTINUE
!RUN DBINSTAL.PUB.DISC                                (Runs DBINSTAL)
SALES                                                  (Specifies the first database)
Z!                                                    (Removes OMNIDEX root file and indexes)
Y                                                    (Answers "yes" to fail-safe prompt)
EXIT                                                  (Exits to the next database)
ACCTS                                                (Specifies the next database)
Z!
Y
QUIT                                                  (Exits to the operating system)
!IF CIERROR <> 0 THEN
!  SHOWJCW
!  TELL MGR.DATA; ODXOUT: Job stream Failed !!!
!ELSE
!  TELL MGR.DATA; ODXOUT: Job stream completed.
!ENDIF
!IF CIERROR > 1609 AND CIERROR < 1628 THEN
!  COMMENT - TELL Message failed
!  SHOWJCW
!  SETJCW CIERROR=0
!ENDIF
!EOJ
```

Other Performance Considerations

This section discusses things you can do to improve the performance of, and minimize maintenance to OMNIDEX.

Estimating disk space requirements

Disk space is required by OMNIDEX for:

- unloading keywords during an indexing operation
- storing keywords and key values in OMNIDEX indexes

During an indexing operation, you should provide enough disk space for indexes and unload files. After a reindexing operation, you need only provide enough disk space for the index files.

You can estimate the disk space required for unload files and index files by following the guidelines below. To determine how many keywords are unloaded or indexed per domain, follow the guidelines described under “Number of keywords to be indexed”, on page 4-100.

Keyword index files and unload files

The disk space collectively required by the index files (or unload files) of a given domain depends on:

- the number of keyword keys being indexed
- the number of keywords per key⁵
- the number of unique keywords versus the number of repeating keywords

As a rule of thumb, domains that have default keys installed on linked detail sets (record specific keys) require approximately eight bytes per keyword. The formula to estimate how many sectors are required to index such a domain is:

$$\frac{\text{keywords} \times 8}{256 \text{ bytes per sector}} = \text{sectors}$$

5. For more information about delimiting keywords, see Appendix B.

All other domains require approximately five bytes per keyword. This includes domains that contain:

- no keys on detail sets
- detail sets where most keyword keys use the Record Complex option
- a single, unlinked, Detail Record (DR) indexed detail set

The formula to determine how many sectors are required to index such a domain is:

$$\frac{\text{keywords} \times 5}{256 \text{ bytes per sector}} = \text{sectors}$$

When indexing keyword values, unload files hold keywords before they are sorted and loaded into the indexes. The unload files for a given OMNIDEX domain requires approximately the same space as the index file. Therefore, to determine the number of sectors required to index a domain, multiply the index file size by two.

Sorted key index files

The disk space required for each sorted key depends on:

- the length of the key, including the key field or composite key length, and the length of the SI⁶
- the amount of data compression that can be performed on the key values⁷
- the number of records in the set

The formula used to estimate how many sectors are required to index such a domain is shown below. Note that the formula is multiplied by 1/3 to account for index compression and that the SI length for any detail set is 4 bytes for the relative record number.

$$\frac{1}{3} \times \frac{\text{records} \times (\text{key length} + \text{SI length})}{256 \text{ bytes per sector}} = \text{sectors}$$

To calculate the amount of disk space required to index a given data set or database, simply combine the individual disk space requirements for each key.

-
6. For detail sets, this is always 4, for the length in bytes of the relative record number.
 7. The amount of compression may vary depending on the uniqueness of the key values being indexed (the more unique, the less compression).

IMAGE-only batch updates

When performing large batch updates that affect more than 25 percent of the database, you should disable the database for TPI, as discussed on page 4-83, and do either of two things.

- ❑ Use a program that calls DBIPUT, DBDELETE and DBIUPDATE in IMAGE-only mode.
- ❑ Use a program that calls TurboIMAGE update intrinsics and run it without Call Conversion.

Because neither method updates the indexes, performance is better than for normal mode updates.

Then, reindex the sets (or domains) where the batch updates were performed. See “Reindexing”, on page 4-78 of this chapter, and “Loading the Indexes”, on page 2-28, for more information. You can perform deferred or batch updates off-hours, during periods of low system activity.

See chapters 2 and 3 of the *OMNIDEX ImagePlus SDK API Guide* for information about IMAGE-only updates.

Disabling Third Party Indexing for maintenance operations

When changing set capacities, if you do not use DBMGR or another TPI-aware utility (like Adager) to change capacities, you should disable Third Party Indexing for a database. When databases are enabled for Third Party Indexing (TPI), their OMNIDEX indexes are updated automatically by TurboIMAGE/iX intrinsics as TurboIMAGE data is updated.

Normally, this feature does not pose performance issues. But when you change set capacities or reload sets in an OMNIDEX database that is enabled for TPI, the automatic updating of indexes can create a serious performance bottleneck. For example, if a utility calls TurboIMAGE/iX intrinsics to reload 500,000 records in a TPI-enabled database, the resulting operation may delete the key values for each of the 500,000 records from the OMNIDEX indexes, then re-add the key values of each record back to those indexes.

A much faster way to perform set reloads and capacity changes is to disable TPI for the database, as discussed on page 4-81, before performing the reload or capacity change. Then, use OmniUtil to reindex the reloaded set. Finally, re-enable the database for TPI.

Tuning indexing performance

When performing an indexing or reindexing operation with OmniUtil, you may be able to optimize your time and system resources by setting the index buffer size. To do this, select 3. Set Index Buffer Size from the Reindexing Options submenu before you reindex or generate the reindexing job.

The acceptable values for index buffer size range from 2 to 128 (in increments of 2) megabytes of memory. The default buffer size is 8 megabytes, which is suitable for database's with up to 100 million keywords.

Setting the index buffer size improves indexing performance in two instances.

- ❑ When indexing small databases, reducing the index buffer size reduces the memory required.
- ❑ When indexing large databases, increasing the index buffer size decreases the time an index operation takes.



Always leave at least 32, megabytes of your system's physical memory for system tables and file mapping.

Below are some rules of thumb for setting index buffer size.

- ❑ When indexing databases that contain between 1 million and 100 million keywords, do not set the index buffer size. Use the default setting of 8 megabytes.
- ❑ When indexing databases that contain 1 million keywords or less, set the index buffer size to the minimum, 2 megabytes, to conserve memory.
- ❑ When indexing databases that contain 100 million keywords or more, set the index buffer size to 32, 64 or 128 megabytes to speed up the indexing process. Leave at least 32 megabytes of memory available for system resources.

The factors that determine the best index buffer size are discussed next.

Number of keywords to be indexed

The number of keywords a domain contains depends on the number of keyword keys installed on it, the average number of keywords per keyed column (or field), and the number of records per table. To estimate the number of keywords in a database, perform the following calculations:

1. To determine the average number of keywords per record for the first table (data set), total the number of words contained in the keyed fields of five records. Divide this total by five.
2. To determine the total number of keywords for the table, multiply the average number of keywords per record (from step 1), by the number of record entries for that table.
3. Repeat steps 1 and 2 above for each table in the domain. Combine the totals for all tables in the domain.

The only way to exactly determine how many keywords are contained within a domain is to see how many keywords are unloaded during an indexing operation. You can check this by running OmniUtil in trace mode, like this:

```
RUN OMNIUTIL.PUB.DISC;INFO="-T"
```

When you perform an indexing operation, check the output window after all the sets in a domain have been unloaded.

Similarly, you can examine the output file in OUT.HPSPOOL after streaming an indexing job.

In either case, the output looks something like this:

```
Unloading keywords
CUSTOMERS          1000 records processed
ORDER-LINES        232 records processed
CUSTOMER-NOTES     3336 records processed
49726 keywords unloaded
```

Use of memory

If you are indexing an OMNIDEX database during off hours, when more memory is available, increasing the index buffer size won't adversely affect other system users. During business hours, however, when memory is in greater demand, you may want to reduce the index buffer size when indexing small databases with few (one million or less) keywords.

Index larger databases at times when more memory is available, and specify a larger index buffer size. This reduces the time required for an indexing operation. You should always leave at least 16 (preferably 32) megabytes of your system's physical memory available when setting the index buffer size. This leaves enough room for system tables and file mapping, and prevents excessive disk swapping.

Re-specifying excluded words

You should periodically check and revise the excluded words list. Exclude keywords that occur too frequently to be useful in searches (like "Inc"), and keywords that have no meaning in searches (like "the"). Excluding keywords from indexing minimizes the space required for the keyword key indexes, and increases update and reindexing performance.

To see what keywords are indexed and how often they occur, run DataView, open the database, and access the data set that contains the most records. Then, perform a keyword search against a textual key, using the search argument !0:Z. This performs a keyword-only search, which can take a while for large domains with many keywords. DataView lists the indexed keywords. For each keyword, DataView shows the number of records that contain it.

See "Excluding Words from Indexing", on page 2-22 for more information about excluding keywords from indexing.

Streamlining OMNIDEX Databases

This section discusses some changes you can make to your existing database to streamline it for use with OMNIDEX. Although these changes are not required, they may benefit you by decreasing the amount of disk space your database requires and improving the performance of your database.

“Possible modifications”, below, discusses some of the changes you can make to your database to improve performance.

“Changing database structure”, on page 4-107, tells you how to revise and compile a schema, create the new database, and transfer data and ID values to it.

Possible modifications

Below are some of the possible modifications you can make to an existing database to improve performance.

- ❑ Eliminate automatic masters and replace their paths with keyword keys.
- ❑ Eliminate sort paths and KSAM files, and replace them with composite sorted keys.
- ❑ Change detail sets into manual masters if they contain unique information. Then, install keyword keys on them to support multiple field access.
- ❑ Add integer search item (SI) fields to your data sets to speed data retrieval and eliminate extra indexes.

Each of these structural changes is discussed below. Procedures to implement these changes are also discussed. Sometimes, changes must be made by revising and recompiling the schema. Other changes can be made with a database utility. When both means are possible, both are discussed.

Note that certain changes may require reinstallation, reindexing, or both. See “Maintenance”, on page 4-68, for more information.

Eliminating automatic masters

OMNIDEX keys cannot improve on the performance of a full chained read. However, to use partial-key values, or perform partial or sorted chained reads, you could replace an automatic master key with a sorted key.

This would require changes to existing programs that call DBGET using a master's SI value. Instead, applications should call DBFIND under TPI, or DBIFIND under ImagePlus to use OMNIDEX keys to find records.

Many database utilities, including Adager, provide a command (PATHDEL) to delete TurboIMAGE paths and data sets. When you have installed OMNIDEX keys on any fields that were accessed by automatic masters, you can eliminate those automatic masters.

Eliminating automatic masters without a utility involves eliminating them from the database's schema, recompiling it, recreating the database and transferring data. These procedures are discussed in "Changing database structure", on page 4-107. You may also need to revise OMNIDEX programs you have written (modifying the mode for DBFINDs, for example). For information about OMNIDEX programs and intrinsics, see the *OMNIDEX ImagePlus SDK API Guide*.

4

Eliminating sorted paths

Sort paths may require much update overhead to maintain, especially for large data sets. Replace them with composite sorted keys that combine the sort field with the SI. This supports the listing of records in a sorted order, without the need to maintain pointers in the data set.



If you replace the TurboIMAGE key with a sorted or keyword key, the chronological order of records added to a detail chain is not maintained.

You can use a database utility (like Adager's DELETE SORT command) to delete sorted paths.

To eliminate sorted paths without a utility, revise the appropriate path statement in the database's schema, recompile the schema, recreate the database, and transfer data to the restructured database.

Eliminating KSAM files

When sorted keys are installed on your database, you may be able to purge some KSAM files to recover disk space. Use a file manager utility to do this.

Changing detail sets into manual masters

Often, logical master sets are created as TurboIMAGE detail sets to permit access by several keys. Because OMNIDEX provides keyword access to master sets, you may want to redefine such sets as masters in the database's schema. This is especially true when redefining a set lets you eliminate automatic masters. To redefine a detail set as a master, you must revise the database's schema and recompile it. Then, recreate the database using DBUTIL. See "Changing database structure", on page 4-107, for more information.

Adding an integer search item (SI)

As discussed in the database design chapter, integer (I2, J2 and K2) search items (SIs) are the most efficient TurboIMAGE SIs. In an OMNIDEX-enhanced database, integer search items (SI/IDs) speed access to data and eliminate additional indexes. Eliminating additional indexes results in better update performance.

Because fast access to a master data set in an OMNIDEX-enhanced database no longer depends on your users knowing the SI value, an integer SI does not pose any retrieval problems for your users.

Most database utilities provide a function to add items to a data set, Adager's ITEMADD, for example. After you add an integer SI to a master and its corresponding detail sets, use DISC's IDADD utility to load the new SI/ID field in the master and its details. IDADD is discussed in "Add ID values", on page 4-111.

If you don't have a utility, you can add an integer SI field to a master and its details by modifying the schema. First, add the new integer SI to the item list and the appropriate data sets. If you will use the old character SIs as a reference when assigning the new SI/ID values, leave them in the schema. Modify the path statements of any affected data sets to reference the new integer SIs. Finally, recompile the schema and create the database as described in "Changing database structure", on page 4-107.

Logical design changes

Changing details to masters

TurboIMAGE's two-level structure of master sets and detail sets often poses design problems. OMNIDEX's Multifind capability offers solutions to these problems that are considerably more streamlined than TurboIMAGE's solutions.

The two most notable logical constructs that pose physical problems in database design are hierarchies greater than two levels, and a many-to-many correspondence. The relationships, their TurboIMAGE solutions, and their OMNIDEX Multifind solutions are discussed next.

Hierarchies greater than two levels

Often, in normalizing a database, the logical hierarchy of records can extend past two levels. For example, a customer may place several orders over the course of business. Each one of those orders consists of individual items: some may be shipped, while others are back-ordered. Based on the logical model above, the physical arrangement of data sets would be a CUSTOMERS master, with many ORDERS detail records, each of which can have many ORDER-LINES detail records. The set hierarchy should look like Figure 4-1 below.

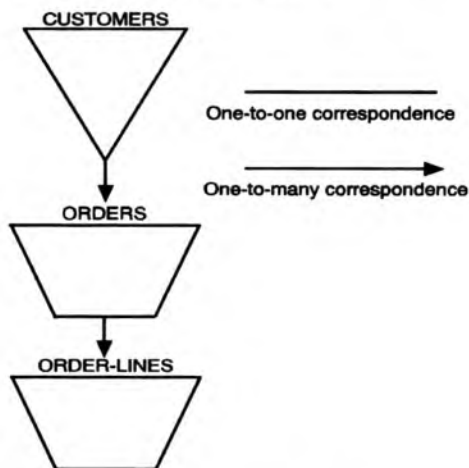


Figure 4-1: A multilevel hierarchy

TurboIMAGE is physically arranged in two levels. The only way to construct a hierarchy greater than two levels is to use automatic masters to serve as cross reference sets between the details which represent the third through *n*th levels. This construct is represented in Figure 4-2.

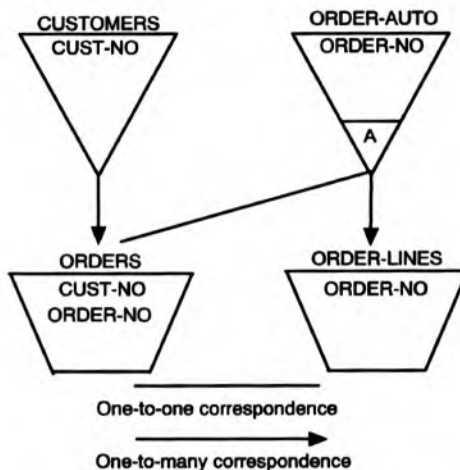


Figure 4-2: A multilevel hierarchy in TurboIMAGE

Because Multifind will permit a master to access another master directly, it can simulate a multilevel hierarchy. You can create a logical path between masters by installing the search item of the level 1 master as a keyword key⁸ on the level 2 master. You could then access the level 2 master from the level 1 master. By the same means the level 2 master could access the level 3 master, which could access the level 4 master, and so on down to the last level of the hierarchy, which would be a detail record. This is shown in Figure 4-3 on the following page.

As you can see, the Multifind approach is more streamlined with respect to eliminating the intermediary automatic masters. The SALES database in DEMODB.DISC was designed before Multifind. The ORDER-MASTER set could have included a keyword key for CUSTOMER-NO, to permit direct access from CUSTOMERS. ORDER-LINES, then, could have been included in ORDER-MASTER's SI domain. In SALES, as it is now, you can access CUSTOMERS from ORDER-MASTER via ORDER-LINES.

8. Install the No Parse option (see page 2-10) to ensure that the entire SI value is preserved and not parsed into keywords.

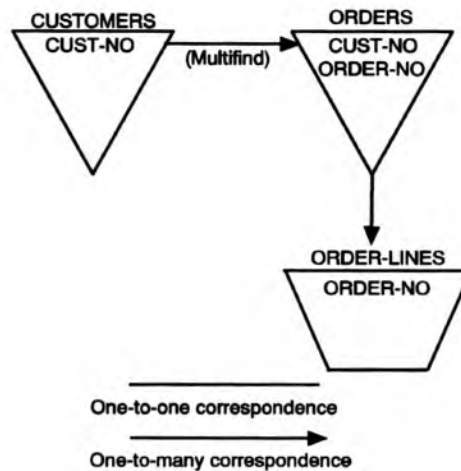


Figure 4-3: A multilevel hierarchy in OMNIDEX.

Changing database structure

4

Some structural changes, like redefining a detail set as a master set, must be made to the database by revising and recompiling its schema. Other structural changes, like deleting automatic master sets, can be made using a utility, but may lead to situations where streamlining of the database structure is possible only through major structural changes.

This section is designed to help you implement structural changes to your database if:

- you don't have access to a database utility.
- you are going to make major structural changes.
- you are designing a new database to best use OMNIDEX's many features.

The steps to making structural changes are discussed below.

1. Revise and compile the schema.
2. Create the new database.
3. Load data, including OMNIDEX ID values. This step may require converting the data to fit the new structure.

Step 1: Revising and compiling the schema

You can revise the schema source file for a database using any EDIT/3000 compatible text editor. If you would like to practice revising a schema, use the file STORESC.DEMODB.DISC. Keep the revised file under a different name to preserve the original.

After you have edited the schema to reflect any changes, you must recompile it using DBSCHEMA. To recompile the schema, follow the steps below:

1. Log on to the group and account where you want the database to reside. In this case, log on to the DEMODB group of the DISC account.
2. Compile the schema by running DBSCHEMA and supplying the STDIN parameter, like this:

```
RUN DBSCHEMA.PUB.SYS;STDIN=STORESC.DEMODB.DISC
```

Or you can use file equates before running DBSCHEMA:

```
FILE DBSTEXT=STORESC.DEMODB.DISC  
FILE DBSLIST;DEV=LP  
RUN DBSCHEMA.PUB.SYS;PARAM=3
```

Step 2: Creating the modified database

Be sure you are still logged on to the group and account where you want the database to reside, in this case, DEMODB.DISC. Use DBUTIL to create the database. At an MPE system prompt, enter:

```
RUN DBUTIL.PUB.SYS,CREATE
```

Enter your database's name at the Database name: prompt. The database is created in a normal TurboIMAGE environment.

```
Database name: STORE
```

When the revised database has been successfully created, you can transfer data from the old database to the new one, as discussed next.

Step 3: Transferring data

You can copy data directly from a data set in one database to a corresponding data set in another database using a DATADEX XFER operation. You also can use an XFER operation to copy data from a database to MPE files. Then, you can use a DATADEX BUILD,IMAGE operation to load the data into the revised database.

A DATADEX XFER operation works if either of two conditions are met.

- If all fields are identical in the new and the old version of the data set.
- If any new fields were added to the end of the data set.

If neither of these conditions are met, convert the data to match the new data set structure, before you load the data into the new data set.

If you added an I2, J2 or K2 field as the OMNIDEX SI/ID of a master set, you must add the ID values before loading the data. Use IDADD, a utility provided with OMNIDEX, to do this.

If you made any structural changes like adding items to the middle of a set, you must convert the data in the old database to accommodate the structure of the new database. You may be able to use utilities to convert the data, like SUPRTOOL from Robelle Consulting, or QTP from Cognos Corporation.

To convert data and add OMNIDEX SI/ID values, follow the steps below.

1. Copy the data from the old database to MPE files.
2. Convert data to the new data set structure.
3. Add OMNIDEX ID values.
4. Load the data into the new database.

These steps are detailed on the following pages.

Copying data to files

The first step in converting a database is to copy the data from the old database to MPE files. There are several possible ways to do this:

- a DATADEX XFER operation
- a QUERY / 3000 SAVE operation
- Dictionary / 3000's DBUNLOAD and DBLOAD commands

The DATADEX XFER operation can copy data from a data set to an MPE file in the same format as the original database record. To copy all the records from a data set to an MPE file, run DATADEX and specify the name of the source database and the data set.

For a database like STORE, the steps would be:

1. Run DATADEX.
RUN DATADEX.PUB.DISC;LIB=P
2. Enter the database's name, the password, and an open mode.
Database name? STORE
Password? DO-ALL
mode? 1
3. Access a data set. Enter the name, or number, of a data set at the Data set? prompt.
Data set? CUSTOMER (or 1 for the data set number)
4. At the DATADEX Command? prompt, use the XFER command (X) to transfer all (@) the records in the current set serially (SERIAL), as follows.
Command? X @,SERIAL
5. When prompted for a destination for the data, enter a file name. Enter M (for MPE) followed by part of the data set name. If the output file does not exist yet, follow its name with ;NEW.
For example, for CUSTOMER, you could enter MCUST;NEW as the destination:
Destination? MCUST;NEW
6. After the data transfer is complete, access the second data set (PRODUCT) by entering E followed by the set name or number (E PRODUCT or E 2). Then enter X@,SERIAL, and specify another destination file (like MPRODS;NEW). Repeat the steps above for each data set in the database.
7. Finally, sort each file by its SI using a program like Hewlett-Packard's SORT or Robelle Consulting's SUPRTOOL. Sort each detail chain by its OMNIDEX SI (if it will be linked to a master) and any secondary sort fields. Name the output file for the MPE file, plus SM for master sets, or SD for detail sets (like MCUSTSM, for CUSTOMER).

Convert the data

Next, convert the data to the new data set structure. If the only change you made was to add an I2, J2 or K2 field as the Search Item, then all you must do is add the values for the OMNIDEX ID.

You can do this with the IDADD utility, which resides in UTIL.DISC. Note that you could also use QTP, or your own program, to add the OMNIDEX SI/ID values.

Add ID values

If you add an integer SI as an OMNIDEX SI/ID to a master data set, you must supply values for the IDs before you can load the data in the MPE files into the new database.

You must also supply OMNIDEX SI/ID values for any associated detail sets that were linked to that OMNIDEX master by the old character SI. In addition, you must also maintain the relationship between the master records and their associated detail data set records.

If the ACCOUNT SI field of the CUSTOMER master in the STORE database was an X10 field, you may want to add a J2 search item, ODX-ID, for use as an OMNIDEX SI/ID to the CUSTOMER master and SALES detail data sets. If you did this, you would need to assign the same ODX-ID value to all records that shared the same ACCOUNT value.

The data set structure for the CUSTOMER master and its associated detail SALES are shown below:

```

NAME:  CUSTOMER,    MANUAL
ENTRY:  ODX-ID(1),
        ACCOUNT,                <<sorted>>
        LAST-NAME,              <<sorted & keyword>>
        FIRST-NAME,             <<keyword>>
        INITIAL,
        STREET-ADDRESS,
        CITY,                    <<keyword>>
        STATE,                   <<keyword>>
        ZIP,                      <<keyword>>

```

```

NAME: SALES,          DETAIL
ENTRY: ODX-ID        (CUSTOMER),
                   ACCOUNT,          <<composite sorted key >>
                                     <<with PURCH-DATE >>
                   STOCK# (PRODUCT),  <<keyword>>
                   QUANTITY,
                   PRICE,
                   TAX,
                   TOTAL,
                   PURCH-DATE,        <<keyword>>
                   DELIV-DATE;        <<keyword>>

```

Instead of the ACCOUNT field, the ODX-ID field is used as the path between CUSTOMER and SALES, and the same correspondence between the data sets would need to be maintained.

For example, if the value of the ODX-ID for a given CUSTOMER record is 10, and the ACCOUNT number value is 129CO, the value of the ODX-ID in the corresponding SALES records (those with ACCOUNT number 129CO) should also be 10.

Master Record

```

ODX-ID           -> 10
ACCOUNT          -> 129CO
LAST-NAME        -> Packard
FIRST-NAME       -> Hewlett
STREET-ADDRESS   -> 999 Pruneridge Ave.
CITY             -> Cupertino
STATE           -> CA
ZIP             -> 95014

```

Detail Records

ODX-ID	-> 10
ACCOUNT	-> 129CO
STOCK#	-> 1042ABNZ6
QUANTITY	-> 15
PRICE	-> 100.00
TAX	-> 90.00
TOTAL	-> 1590.00
PURCH-DATE	-> 890117
DELIV-DATE	-> 890124
ODX-ID	-> 10
ACCOUNT	-> 129CO
STOCK#	-> 1044N56
QUANTITY	-> 1
PRICE	-> 4550.00
TAX	-> 250.00
TOTAL	-> 4700.00
PURCH-DATE	-> 890120
DELIV-DATE	-> 890601

IDADD is a utility provided in the UTIL group of the DISC account. It reads sorted MPE files and inserts an OMNIDEX ID value in the master and detail records so that their relationship is maintained.



MPE files must be sorted by the old SI values before using IDADD to add the OMNIDEX IDs.

After the files are sorted, run IDADD. A sample execution of this program for a CUSTOMER master and a SALES detail data set, like those in the STORE database, is shown below.

```
RUN IDADD.UTIL.DISC
```

```
DS76004.2.09 IDADD WED, FEB 11, 1989, 10:16 AM (C)
Dynamic Information System Corp. 1984
```

General Information? (Y/N) Y

This program adds a double integer ID value to each record in all of the files you specify. The ID value is placed at the beginning of the record (position 1) unless you override this default by appending a ; to the name of the output file. For example, OUTFILE;33 inserts the ID into byte positions 33-36 of the record. The ID values assigned start at 1 and increment by 1 for each master record.

All files must be sorted. All detail records with the same key value as a master record are assigned the same ID value as that master record. If a detail record has a key value that does not match the key value of any master record, that detail record is discarded and its record number is displayed.

IDADD can be run with the following PARM values:

PARM = 1 ; Expects a Master File only, sorted or not

PARM = 2 ; Lets you change the starting ID value

PARM = 3 ; Both 1 & 2 above

PARM = 4 ; Lets you reassign the existing ID values

Enter // to stop

Master file: **MCUSTSM** (file containing sorted master records)
Key position (1..256)? 1 (starting position of old TurboIMAGE key)
Key length in bytes? 10 (key length of old TurboIMAGE key)
Output file: **IDMCUST** (new file with inserted OMNIDEX IDs)

Detail file: **MSALESD** (file containing sorted detail records)
Key position (1..80)? 1 (starting position of old TurboIMAGE key)
Output file: **IDSALESD** (new file with inserted OMNIDEX IDs)
Detail file: // (additional details could be specified)

Are all input files sorted (Y/N)? Y

2 files have been designated for conversion. Proceed? Y



-
-
1. If the old TurboIMAGE SI is not at the beginning of the record, you simply specify its position at the `key position (1..256)?` prompt.
 2. If you are adding OMNIDEX ID values for a detail set that has an automatic master as its OMNIDEX master, specify the detail set as a master set.
 3. You can place the OMNIDEX ID values anywhere in the record for the output file by specifying the output file name plus a semicolon (;) and the correct position number. For example, if you are adding the OMNIDEX ID to the end of a 78 byte detail record, you would enter `IDDETAIL;79` at the `Output file:` prompt.
 4. If a data set was a detail in the old database, but is a master set in the new database, specify it as a master during IDADD.
 5. If a master set has no details related to it, you can use `PARM=1` when running IDADD. In this case, you need not sort the master file. This parm also lets you add ID values to a file that does not contain unique SI values.
-
-

Use the output files created with IDADD to load data into the new database, as described next.

Load the data from files

The fourth step in converting a database is to load the data from the MPE files. To do this, you can use the DATADEX BUILD,IMAGE command.

To load the data for a database, like STORE, you would do the following:

1. Run DATADEX.

```
RUN DATADEX.PUB.DISC;LIB=P
```

2. DATADEX prompts you for a database name. Enter your database name, for example, STORE.

When prompted for the password, enter the password. For STORE, it is DO-ALL.

When prompted for the mode, enter 3 to open the database with exclusive access. This loads the files 3 to 5 times faster than mode 1.

3. DATADEX asks for the set you want to work with. Enter ? to see a list of the data sets.

Press **[return]** to return to the `Data set?` prompt. Then enter the name or number of a data set, like the CUSTOMER data set in the STORE database. Be sure to load the master data sets before loading their associated detail sets.

4. You are prompted for a command. Enter BUILD,IMAGE to load the data from an external MPE file. The IMAGE option loads records into data sets without indexing them. You can index records faster using an OmniUtil indexing operation.

If you have less than 500 records to index, you could use the DATADEX BUILD command alone to load and index the data sets.

5. DATADEX asks for the name of the file that contains the data to be loaded into the data set.

Enter the file name, in this case IDMCUST. DATADEX reads the IDMCUST data file and copies it into the CUSTOMER data set.

6. DATADEX asks for another command. Enter E 2 to switch access to the second data set.

Enter BUILD,IMAGE again at the `Command?` prompt. Then enter the next file name at the `Source file?` prompt to load the data into the second data set.

Repeat these steps for each data set in the database.

Appendix A: Data Type Considerations

You can key almost any field for keyword or sorted access. Integer fields longer than four words are not supported for keyword access. Some minor restrictions apply to zoned (type Z), real (type R and E), and packed (type P) fields.

Integer fields (Types I, J or K)

You can install both keyword and sorted keys on type I, J or K fields, subject to the following restrictions.

- ❑ You can't install keyword keys on fields greater than four words (greater than I4, J4 or K4).
- ❑ Partial (generic) argument values are not supported in searches, but ranges are.
- ❑ You can group integer keys only with other integer keys. Grouping is discussed on page 2-9 of the "OMNIDEX Installation" chapter.
- ❑ The correct collation of range searches on keyword keys is supported in OMNIDEX version 3.0 and later. The correct collation of sorted key searches is supported in OMNIDEX version 3.1 and later.



Zoned fields (Type Z)

You can install both keyword and sorted keys on type Z fields. Zoned fields are ASCII fields that support signed numbers. The absolute value of the number is stored in all but the last byte, which is reserved for an overpunch character that represents the sign. For example, a signed Z6 field contains the following value for "2":

```
00000B
```

If you enter the argument "2", an application program must convert it to "00000B" before searching. Otherwise, the search intrinsic will look for the value "2" and never find it.

Data returned to an application must be converted back to a form that is suitable for display.

The only restriction for zoned keys applies to versions of OMNIDEX before 3.2 where the correct collation of range and sorted search operations is not supported due to the effect of the overpunch character on indexing. In OMNIDEX version 3.2 and later, zoned key values are indexed in true numeric order from the lowest negative value to the highest positive value, and therefore support ranges and sorted access..

Packed fields (Type P)

You can install both keyword and sorted keys on type P fields, subject to the following restrictions.

- You can't use partial (generic) values in keyword or sorted searches.
- The correct collation of range searches on keyword keys is supported in OMNIDEX version 3.0 and later. The correct collation of sorted searches is supported in OMNIDEX version 3.1 and later.
- You can group packed keys only with other packed keys. See "Grouping", on page 2-9 of the "OMNIDEX Installation" chapter.

In sorted retrievals on packed keys, the application must convert search values to packed decimal notation before searching. For keyword keys, search values are converted automatically.

Real fields (Type R and E)

You can install both keyword and sorted keys on R2, R4, E2 and E4 fields, subject to the following restrictions.

- ❑ You can't use partial (generic) values in keyword or sorted searches.
- ❑ The correct collation of range searches on keyword keys is supported in OMNIDEX version 3.0 and later. The correct collation of sorted key searches is supported in OMNIDEX version 3.1 and later.
- ❑ You can group real keys only with other real keys. Grouping is discussed on page 2-9 of the "OMNIDEX Installation" chapter.

For sorted keys, an application program must convert any search values to HP real or IEEE real format before using them in a retrieval. For keyword keys on real fields this conversion is done automatically.

Character fields (Types U or X)

You can install keyword and sorted keys on type U or X fields. No restrictions apply. Note that alphabetic values are upshifted before they are indexed. You can use the No Translate option to prevent this, however.

TurboIMAGE field arrays (compound items)

You can install both keyword and sorted keys on compound items. When you install a keyword key on a TurboIMAGE array, the elements of the array are treated as grouped keys by default. To index an array as one big field, use the Blob Indexing option, as discussed on page 2-8 of the "OMNIDEX Installation" chapter.

Note that you must observe word boundaries when isolating parts of binary fields or elements of a binary array into composite keys.



Alternate data types

In some cases, the data type specified by the TurboIMAGE item definition may not reflect the type of data stored in that item for any given record. For example, the ASK software package and PowerHouse 4GL each have their own data type for dates.

In other cases, applications may store data of any type in a field, regardless of its TurboIMAGE type. If the stored data type does not match the TurboIMAGE data type (as defined in the schema), then a *data type discrepancy* exists. For example, an integer array of 13 double word elements (TurboIMAGE type 13I2) may be stored in a TurboIMAGE item of character type (TurboIMAGE type X52). This discrepancy may be resolved in an application program, but can cause problems with OMNIDEX indexing.

Each alternate data type is discussed next.

ASK and PowerHouse date fields

You can install both keyword and sorted keys on ASK or PowerHouse date fields.

Because they are integer fields, the same restrictions apply as those described on page A-1. Any ASK, PHDATE or JDATE fields of data type I or J must be type cast as type K to be supported by OMNIDEX. Type Casting is discussed on page 2-11 of the "OMNIDEX Installation" chapter.

During sorted retrievals, application programs must convert the search date value to the correct binary integer value using ASK Computer System's or PowerHouse's date algorithm. OmniQuest and OmniWindow have features to make such conversions automatically.

For keyword searches that use ODXFIND mode 3 or 5, or DBFIND mode 12 or 22, you can use the %DATE function to convert the date automatically. The %DATE function is passed in the *keywords* parameter. See the "ODXFIND" section of the "Intrinsics" chapter in the *OMNIDEX ImagePlus SDK API Guide* for more information about the %DATE function.

Data type discrepancies

You can install both keyword and sorted keys on a given field, even if the format in which the data is stored disagrees with the TurboIMAGE data type.

Any restrictions that may apply are determined by the stored data type and not the TurboIMAGE data type.

As with other data types described earlier, you must convert a search value to its stored format before using it in a sorted retrieval. For example, if data is stored as a packed decimal type in an ASCII field, you must convert the search argument to packed decimal format before searching.

Because OMNIDEX translates ASCII search arguments automatically, based on the TurboIMAGE data type, you must specify the stored data type during installation. This is done with the Type Casting option in OmniUtil. See "Type Casting", on page 2-11 of the "OMNIDEX Installation" chapter for more information about Type Casting.



Appendix B: OMNIDEX Parsing Rules

OMNIDEX parses the values indexed for both sorted and keyword keys, as well as the arguments entered in a keyword search.¹ This section discusses:

- ❑ how data is parsed during indexing for both types of keys (below)
- ❑ how special characters are parsed during keyword searches (page B-5)
- ❑ how OMNIDEX translates 8-bit extended ASCII characters (page B-8)

Parsing during indexing

Keyword key values

During indexing, OMNIDEX uses spaces and other special characters to parse individual keywords out of a field's contents. If a field contains special characters, you should consider how its data will be indexed. To prevent the parsing of special characters for a given keyword key, install the No Parse option on it. Note that for No Parse keys, the first 32 bytes of a keyed field are indexed as a single keyword.

1. With the exception of wildcard characters (#, ?, and @) in TPI-based DBFIND searches, arguments are not parsed in searches against sorted keys.

Upshifting

Upshifting means converting words to uppercase (capital) letters. This is done for keyword keys so that keyword arguments can qualify indexed keywords, no matter what case they are entered in.

For example, "Dynamic Information Systems Corporation" would be parsed and indexed as four separate words because each word is separated by a blank. Each word would be stored in capital letters as "DYNAMIC", "INFORMATION", "SYSTEMS" and "CORPORATION".

Keywords are upshifted in the indexes and during searches, unless the No Translate option has been specified for that field. No Translate is discussed on page 2-15 of the "OMNIDEX Installation" chapter.

Excluded words

Every value in a keyword key is indexed and can be used for keyword retrieval, unless it is in the *excluded word list*.

Excluded words are words or ASCII numbers that do not need to be indexed because they are not useful for retrieval. The keyword "Inc" is a good example. By using OmniUtil, you can specify an excluded word list for each database. The process for excluding words is discussed in "Excluding Words from Indexing", on page 2-22 of the "OMNIDEX Installation" chapter.

You can disable the excluded words list for certain fields with the No Exclude option. No Exclude is discussed on page 2-15 of the "OMNIDEX Installation" chapter.

Spaces and nulls

A field that contains all spaces or null characters is not keyworded unless the No Parse and No Exclude keyword options are selected. No Parse and No Exclude are discussed in "Adding key options" on page 2-7 of the "OMNIDEX Installation" chapter.

Special characters

During indexing, most special characters are parsed as spaces that separate the keywords. These special characters are:

! ? * : ; { } [] . , + = " ' ' | ()

Some of these special characters are reserved, that is, they have a special meaning during searches. They are discussed in “Reserved characters and strings” on page B-6.

Generally, a string that contains special characters is indexed as several separate keywords. For example, “Leonard,Feinstein&Fitch” would be indexed as three separate keywords: “LEONARD” “FEINSTEIN” “FITCH”. Here, the comma and ampersand are treated as spaces.

This rule has the following exceptions.

- ❑ If the No Parse option is installed on a keyword key field, the entire field up to 32 characters is indexed as one keyword, including special characters.
- ❑ Five special characters (# \$ % & _) are treated as spaces if they occur at the beginning of a string, but are preserved if they occur in the middle or at the end of a string.

Keyword value	Parsed as
#15	15
15%	15%

Table B-1: Parsing # \$ % & _

- ❑ Commas in numbers are ignored and discarded if they occur on a 3-digit boundary; otherwise they are treated as spaces.

Keyword value	Parsed as
3,800	3800
300,700	300700
3,7	3 and 7
300,370,3388	300370 and 3388
35,3800,80	35 and 3800 and 80

Table B-2: Parsing commas in numbers

B

- ❑ Decimal points (.) between ASCII numbers (data type X) are preserved.

Keyword value	Parsed as
45.0	45.0

Table B-3: Parsing ASCII decimals in numeric strings

- ❑ Decimal points (.) between character and ASCII numeric strings are preserved only if the value starts with the character string. Otherwise they are treated as blanks.

Keyword value	Parsed as
ABC.55	ABC.55
abc.55.50	ABC.55.50
55.ABC	55 and ABC

Table B-4: Parsing ASCII decimals in alphanumeric strings

- ❑ Hyphens (-), slashes (/), and backslashes (\) between characters are preserved, and cause multiple keywording of the values they separate.²

Keyword value	Parsed as
HP\2626	HP\2626 and HP and 2626
1/a	1/A and A and 1
Hewlett-Packard	HEWLETT-PACKARD and HEWLETT and PACKARD

Table B-5: Parsing hyphens

2. This differs for versions of OMNIDEX prior to 3.3.

Parsing data for sorted keys

Upshifting

ASCII sorted key values are upshifted during indexing. Sorted key arguments are upshifted by retrieval intrinsics during retrieval.

Sorted key values are not upshifted for:

- sorted keys installed with the No Translate (NT) option, which is used to disable upshifting.
- sorted keys installed directly on TurboIMAGE search items.
- binary sorted keys or composite sorted keys that contain a binary component.

Spaces and nulls

Leading spaces or nulls are always indexed. Sorted keys that contain all spaces or nulls are not indexed unless the key uses the No Exclude option, or the key is installed on a TurboIMAGE search item.

Parsing during keyword searches

During searches, OMNIDEX intrinsics parse arguments to determine what character strings are argument values, and which are operators. For example, in the following keyword expression:

robert OR ANDERSEN! AND MANAG@

ODXFIND parses as follows:

robert	is upshifted to match the indexed value "ROBERT".
OR	is a Boolean operator.
ANDERSEN!	is a phonetic (Soundex) argument, as indicated by the trailing exclamation point (!). See "Soundex Phonetic Searches", on page 4-7, for more information.
AND	is a Boolean operator.
MANAG@	is a generic argument, the equivalent of "all words that begin with "MANAG". See "Pattern-matched and generic arguments", on page 4-7, for more information.

B

Upshifting

Arguments entered in a keyword search expression are upshifted to match the key values indexed for a key. If the key being searched is installed with the No Translate option, the argument is not upshifted and is passed in case-sensitive form.

Spaces

In ODXFIND modes 3 and 5, which offer enhanced argument parsing, spaces in a keyword expression are parsed as AND operators unless they enclose an operator. In ODXFIND mode 1, spaces in a keyword expression are ignored.

Special characters

Special characters that do not represent reserved characters (like \$, & and so on) can only be included in the middle or end of a keyword argument value. If special characters begin a keyword argument value, they cause ODXFIND error condition -207 Keywords must start with an alphanumeric character or decimal point.

Reserved characters and strings

The following characters have a special meaning to the ODXFIND and DBFIND search intrinsic.

- ! when used at the end of a keyword argument, it indicates a phonetic (Soundex) search.
when used alone or at the beginning of a range or partial-key (generic) argument, it indicates a keyword-only search.
- @ when used anywhere in a keyword argument, acts as a wildcard that indicates an unspecified string of characters of any length.
- # when used anywhere in a keyword argument, acts as a wildcard that indicates a single unspecified digit (0-9).
- \$ when used at the beginning of a keyword argument, indicates that the keyword is the name of an ID file that contains the arguments for the search. See "External ID files", on page 4-23 of the "Topics" chapter, for more information.

- & when used at the beginning of a keyword argument, indicates that a Multifind search should be performed. See "Multifind", on page 4-63 of the "Topics" chapter, for more information.
- * when used at the beginning of a keyword argument, acts to reload the previous list of qualified records. Typically, it should follow an AND, OR or NOT operator.
- () when used anywhere in a keyword argument, acts to nest an expression. A single parenthesis causes the error `Unbalanced parenthesis in argument`.
- when used in the middle of a keyword expression, acts as a Boolean NOT operator.
- + when used in the middle of a keyword expression, acts as a Boolean OR operator.
- : when used anywhere in a keyword expression, acts as a range operator.
- " when used anywhere in a keyword expression, acts to prevent the parsing of argument values. A single quotation mark causes the error `Unmatched quote`.
- > when used at the beginning of a keyword argument, acts as a "greater than" relational operator.
- > when used at the beginning of a keyword argument, acts as a "less than" relational operator.
- , when used anywhere in a keyword expression, acts as a Boolean AND operator.
- ? when used anywhere in a keyword argument, acts as a wildcard that indicates a single unspecified printable character.

The reserved strings for OMNIDEX are AND, NOT, OR, THRU and TO. These apply to ODXFIND modes 3 and 5, and all DBFIND keyword search modes.

Preventing argument parsing

To prevent the parsing out of spaces and special characters, enclose argument values in double quotes ("). See "Arguments that contain special characters", on page 4-9, for more information.

How OMNIDEX translates 8-bit characters

When OMNIDEX indexes values, it upshifts lower case characters to upper case and translates 8-bit extended ASCII characters to a 7-bit equivalent.

Table B-6 shows the 128 Roman8 extended ASCII characters their position in collating sequence, and how OMNIDEX translates them.

ASCII collating number	Roman8 character	OMNIDEX translation	ASCII collating number	Roman8 character	OMNIDEX translation
160	undefined		182	N	N
161	Á	A	183	ñ	N
162	Â	A	184	i	
163	È	E	185	¿	
164	Ê	E	186	□	
165	Ë	E	187	£	
166	Î	I	188	¥	
167	Ï	I	189	§	
168	´		190	f	
169	`		191	¢	
170	^		192	â	A
171	¨		193	ê	E
172	~		194	ô	O
173	Û	U	195	û	U
174	Ü	U	196	á	A
175	Ł	L	197	é	E
176	ˉ		198	ó	O
177	undefined		199	ú	U
178	undefined		200	à	A
179	°		201	è	E
180	Ç	C	202	ò	O
181	ç	C	203	ù	U
204	ä	A	230	Í	I

Table B-6: The default OMNIDEX translation of 8-bit characters

ASCII collating number	Roman8 character	OMNIDEX translation	ASCII collating number	Roman8 character	OMNIDEX translation
205	ë	E	231	Ó	O
206	ö	O	232	Ò	O
207	ü	U	233	Õ	O
208	À	A	234	õ	O
209	î	I	235	Š	S
210	Ø	O	236	š	S
211	Æ	untranslated	237	Ú	U
212	å	A	238	ÿ	Y
213	í	I	239	ÿ	Y
214	ø	O	240	Ɔ	untranslated
215	œ	Æ	241	Ɔ	Ɔ
216	Ä	A	242		untranslated
217	ì	I	243		untranslated
218	Ö	O	244		untranslated
219	Ü	U	245		untranslated
220	É	E	246	—	untranslated
221	ï	I	247	¼	untranslated
222	ß	untranslated	248	½	untranslated
223	Ë	O	249	ª	untranslated
224	Á	A	250	º	untranslated
225	Ã	A	251	«	untranslated
226	ã	A	252	•	untranslated
227	Ð	D	253	»	untranslated
228	ð	D	254	±	untranslated
229	ì	I	255		untranslated

Table B-6: The default OMNIDEX translation of 8-bit characters

Appendix C: Installing OMNIDEX Procedure Libraries

This appendix discusses the placement of Xls and SLs. The process of placing OMNIDEX Xls and SLs is very straight forward. Which Xls and SLs your program should reference is based on the types of intrinsics your program uses. Any program that references XL.PUB.DISC will run successfully although not as efficiently as when it directly references XLOMNIDX.PUB.SYS or XL.PUB.SYS.



If you are upgrading from version 2 of OMNIDEX, you must purge all existing OMNIDEX procedure libraries, segments, and XL modules before installing the version 3 procedure libraries. There are four procedure libraries. They are listed below in the order that they are discussed.

XLOMNIDX.PUB.SYS	(the OMNIDEX intrinsics library) contains the OMNIDEX intrinsics that access and maintain the OMNIDEX indexes.
XL.PUB.SYS	(the TurboIMAGE/iX library) contains the TurboIMAGE intrinsics that access and update TurboIMAGE data sets.
XL.PUB.DISC	(the Call Conversion library) contains routines that coordinate intrinsic calls to reference either the OMNIDEX intrinsic library or the TurboIMAGE/iX library.
SL.PUB.DISC	(the Intrinsics Switch Stub library) contains routines that convert compatibility mode intrinsic calls to native mode intrinsic calls.



In addition to the four XLs mentioned above, two additional XLs can be used to simplify access to OMNIDEX.

XLSWITCH	can be used in place of XL.PUB.DISC to automatically direct an application to the correct copy of XLOMNIDX, if several are present, for any database.
ODXM210	can be added to XL.PUB.DISC210 (for version 2.10 installations) to allow applications to resolve through XLSWITCH.

The four different intrinsic libraries are discussed next.

The OMNIDEX intrinsic library

The OMNIDEX intrinsic library is shipped as XLOMNIDX.PUB.DISC*vrr*. It accesses and maintains the OMNIDEX indexes. The OMNIDEX intrinsic library also interacts with the TurboIMAGE/iX intrinsics to access and maintain your TurboIMAGE data.

To use the OMNIDEX intrinsic library, simply copy it into PUB.SYS using the MPE COPY command. If you used the SETUP program when you installed the OMNIDEX software, it gave you the option to copy XLOMNIDX to PUB.SYS. This makes the OMNIDEX intrinsics directly accessible to the Call Conversion library, and the Standard TurboIMAGE/iX Interface to Third Party Indexing (TPI).

After installing updated OMNIDEX software, there will be an OMNIDEX intrinsic library in the PUB group of your updated DISC account, in addition to the one in PUB.SYS. Your applications should resolve directly through XLOMNIDX.PUB.SYS if they use TurboIMAGE intrinsics to access TPI-enabled databases. If they use the OMNIDEX intrinsic interface, applications can resolve through XLOMNIDX.PUB.DISC*vrr*, as described in "Using alternate OMNIDEX intrinsic libraries" on page C-8.

The TurboIMAGE/iX intrinsics

The TurboIMAGE intrinsics reside in the system library (XL.PUB.SYS). They access and maintain TurboIMAGE data, no matter which interface (OMNIDEX or TurboIMAGE) a program uses. The TurboIMAGE intrinsics interact with the OMNIDEX intrinsic library (XLOMNIDX.PUB.SYS) to access and maintain the OMNIDEX indexes for any database that is enabled for Third Party Indexing.

Your retrieval applications should reference XL.PUB.DISC if:

- ❑ they interact with the QUICK interface.
- ❑ they already reference XL.PUB.DISC or a copy of XL.PUB.DISC.
- ❑ they perform retrievals using DBFIND's and DBGET's extended TPI modes against a database not enabled for Third Party Indexing.

Your update applications should reference XL.PUB.DISC if:

- ❑ they mix calls to OMNIDEX intrinsics and calls to TurboIMAGE intrinsics.
- ❑ they are TurboIMAGE applications updating an OMNIDEX database that is not enabled for Third Party Indexing and you need the indexes updated. An online order entry application, for example, could reference XL.PUB.DISC to always automatically update indexes.

You can run any OMNIDEX application through XL.PUB.DISC. This may be less efficient than referencing either the OMNIDEX or TurboIMAGE intrinsics library directly.

The Call Conversion library

The Call Conversion library, XL.PUB.DISC, acts like a switchboard, to direct your OMNIDEX applications' intrinsic calls to the OMNIDEX intrinsic library (XLOMNIDX.PUB.SYS). The Call Conversion library combines the functionality of XL.CC (Call Conversion) and XL.CCTPI (the IMAGE Retrieval Interface) from version 2 releases of OMNIDEX.



The Intrinsic Switch Stub library

The Intrinsic Switch Stub library, SL.PUB.DISC, allows your compatibility mode applications to access the native mode intrinsics in XLOMNIDX.PUB.SYS. It does this by converting compatibility mode intrinsic calls into native mode intrinsic calls, and directing them to the Call Conversion library (XL.PUB.DISC).

To use SL.PUB.DISC, copy it and XL.PUB.DISC into your compatibility mode application's account, and run your applications with the appropriate LIB parameter. No matter where you place SL.PUB.DISC, XL.PUB.DISC must reside in the same group as the compatibility mode application. If necessary, you can merge the Intrinsic Switch Stub SL and Call Conversion XL with an existing SL and XL. If an application has an SL in the group where it resides, add the Intrinsic Switch Stub segments, copy the Call Conversion XL, and continue to run the application with Lobed. If you installed the Call Conversion and the Intrinsic Switch Stub libraries or procedures into the PUB group of an application's account, run the application with LIB=P.

The various procedures for copying the Intrinsic Switch Stub libraries are discussed in "Using the Switch Stub SL (SL.PUB.DISC)" on page C-6.

Referencing and placing libraries

This section discusses how to reference or place the procedure libraries that are necessary for OMNIDEX to function. Xls can be placed, but do not require it, as discussed next. For programs to reference SLs, however, they must reside in the same account as the SL, as discussed above.

Referencing Xls

There are several ways that a native mode program can reference an XL.

1. Reference the XL when linking the program. To do this, use the LINK command, after compiling your program with a native mode compiler, to link the program to the XL.

```
LINK $OLDPASS,PROGRAM;XL="XL.GROUP.ACCOUNT,XLOMNIDX.PUB.SYS"
```

Note that PROGRAM represents the name of a compiled program (or object) file. GROUP and ACCOUNT represent the group and account where the program's XL resides.

2. Reference the XL in an "XL=" parameter added to the RUN statement. For example:

```
RUN program,XL="XL.group.account"
```
3. Copy the XL into your application program's account using the MPE COPY command. For example:

```
COPY XL.PUB.DISC
```

Run your applications with the appropriate LIB= parameter.

There are no measures required for your native mode programs to access the TurboIMAGE intrinsics, because they reside in XL.PUB.SYS. The OMNIDEX intrinsic and Call Conversion XLs do require you to reference them. Each XL is discussed next.

The OMNIDEX intrinsic library (XLOMNIDX.PUB.SYS)

Native mode applications that use the OMNIDEX Intrinsic Interface can reference the OMNIDEX intrinsic library using either method 1 or 2 discussed above. Applications that use the Standard TurboIMAGE/iX Interface on a TPI-enabled database automatically use XLOMNIDX.PUB.SYS and therefore do not require any special XL references. The TurboIMAGE intrinsics look for XLOMNIDX in PUB.SYS if they access the OMNIDEX indexes during searches or updates.

The Call Conversion library (XL.PUB.DISC)

The Call Conversion library directs TurboIMAGE or OMNIDEX intrinsic calls to XLOMNIDX.PUB.SYS. As a result, programs that either retrieve from or update OMNIDEX databases function properly if they reference XL.PUB.DISC. This is true whether the programs call TurboIMAGE or OMNIDEX intrinsics to update data or retrieve records, and whether or not the database is enabled for Third Party Indexing (TPI).

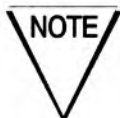
Once an intrinsic call is passed to XLOMNIDX, how it resolves depends on whether or not the database referenced in the *base* parameter is enabled for TPI. If the database is enabled for TPI, the call is passed to the TurboIMAGE intrinsics in XL.PUB.SYS. If the database is not enabled for TPI, the call is processed first by XLOMNIDX.PUB.SYS.

Native mode applications can reference the Call Conversion XL in any of the three ways discussed on page C-4.

Using the Switch Stub SL (SL.PUB.DISC)

To make OMNIDEX available to compatibility mode programs you must copy both SL.PUB.DISC and XL.PUB.DISC (the Call Conversion library) into the account where the program resides. You can copy SL.PUB.DISC into the group where the program resides or into the PUB group of the Program's account. Note that XL.PUB.DISC must reside in the same group as program.

If an SL or XL already resides in the group where you want to install the corresponding libraries, you must merge the Switch Stub segments or Call Conversion executable routines with any existing SL or XL as discussed next. Remember to purge all existing OMNIDEX procedure segments from existing SLs before adding any version 3 segments.



The XL file that contains the Call Conversion executable routines must always be named "XL".

When a compatibility mode program calls a TurboIMAGE or OMNIDEX intrinsic, the routines are resolved in the Switch Stub SL and switched to its native mode counterpart in the Call Conversion XL. The Call Conversion XL then directs the routine to XLOMNIDX.PUB.SYS.

Adding Switch Stub Routines to existing SLs

If there is already an SL in the group where you are installing the Switch Stub and Call Conversion procedures, you can merge them with the existing SL. To add the Switch Stub USL segments to an existing SL, perform the following steps:

1. Log on to the group that contains the SL.
2. Use Hewlett-Packard's SEGMENTER to add the OMNIDEX Intrinsic Switch Stub segments to the existing SL. For example:

```
:SEGMENTER  
  
-SL SL  
-USL USL.PUB.DISC  
-ADDSL DISC'2  
-ADDSL DISC'CM'SWITCH  
-ADDSL DISC'4  
-ADDSL DISC'7
```

```
-ADDSL DISC_C  
-ADDSL CCSCSEG0  
-ADDSL CCSCSEG1  
-ADDSL CCSCSEG2  
-ADDSL UTIL'1  
-ADDSL IMAGE'CC  
-EXIT
```

Adding Call Conversion routines to existing XLs

If there is already an XL in the group where you are installing the Switch Stub and Call Conversion procedures, you can merge them with the existing XL. To add the Call Conversion XL routines to an existing XL, perform the following steps.

1. Log on to the group that contains the XL.
2. Run Hewlett-Packard's Link Editor by typing the LINKEDIT command.

```
LINKEDIT
```
3. Copy the Call Conversion XL into the existing XL using the COPYXL command.

```
LinkED> COPYXL FROM=XL.PUB.DISC;TO=XL
```
4. Exit LINKEDIT.

```
LinkED> EXIT
```



Using alternate OMNIDEX intrinsic libraries

To direct programs to an OMNIDEX intrinsic library other than XLOMNIDX.PUB.SYS, you can do either of two things.

- ❑ Set the DISC_XLNAME environment variable for programs that resolve through the Call Conversion or Switch Stub library.
- ❑ Use the TPIXLCFG command file in UTIL.DISC*vrr* to direct OMNIDEX utilities to the appropriate copy of XLOMNIDX.

Each of these methods is discussed below.

Setting DISC_XLNAME

To direct applications that resolve through the Call Conversion or Switch Stub library to a copy of XLOMNIDX other than the one in PUB.SYS, set the DISC_XLNAME environment variable using the following syntax:

```
SETVAR DISC_XLNAME "XLOMNIDX.group.acct"
```

For example, to direct your programs to XLOMNIDX.PUB.DISC304, you would enter:

```
SETVAR DISC_XLNAME "XLOMNIDX.PUB.DISC304"
```



Setting this environment does not affect programs that reference XLOMNIDX.PUB.SYS in their XL path. Further you cannot redirect TurboIMAGE calls against TPI-enabled databases. DISC_XLNAME affects only programs that reference the OMNIDEX Intrinsic Switch Stub library (SL.PUB.DISC*vrr*) or Call Conversion library (XL.PUB.DISC*vrr*).

If you redirect your programs to a different XLOMNIDX, you should also supply file equations to redirect them to the corresponding ODXERROR and OXTPIMSG error message files. For example, if you redirected your programs to XLOMNIDX.PUB.DISC304, you should also redirect them to the corresponding error message files as shown below:

```
FILE ODXERROR.PUB.SYS=ODXERROR.PUB.DISC304  
FILE OXTPIMSG.PUB.SYS=OXTPIMSG.PUB.DISC304
```

Using the TPIXLCFG command file

The TPIXLCFG command file, which resides in UTIL.DISC, is useful when you receive an OMNIDEX update tape. It lets you configure the newest utilities to work with either TPI-enabled databases, or databases not enabled for TPI.

To use the TPIXLCFG command file, follow these steps.

1. Log onto the DISC account as manager.
2. Enter TPIXLCFG.UTIL at the system prompt.
3. Follow the directions on the configuration menu:

```
This command file alters several DISC utility programs
to use either XLOMNIDX.PUB or XLOMNIDX.PUB.SYS.
```

```
You must be logged on in the DISC account to use this command.
```

```
Please choose which XLOMNIDX you wish the DISC utilities to use:
```

- ```
1) XLOMNIDX.PUB.SYS (If you have ANY TPI enabled databases)
2) XLOMNIDX.PUB (If you have ONLY non-TPI enabled databases)
3) Quit without changing files.
```

```
Enter Option number:
```

Enter the option number that corresponds to the type of databases that you will access.

You can use this command file to change between using XLOMNIDX.PUB.DISC*vr* and XLOMNIDX.PUB.SYS.

As a shortcut, you can append either PUB, to use XLOMNIDX.PUB.DISC*vr*, or PUB.SYS, to use XLOMNIDX.PUB.SYS, to the command file name at run time. For example:

```
:TPIXLCFG PUB.SYS (to use XLOMNIDX.PUB.SYS)
```







---

# Appendix D: OMNIDEX Version 3 Limits

---

This appendix lists some limits and maximums for OMNIDEX version 3.

## Opens per process

The maximum number of databases open per process is from ten to twelve, depending on the number and types of keys and options installed on the databases and the number of excluded words configured for each database.

## Database maximums

### Number of index files

The maximum number of index files per database is 255.

### Number of OMNIDEX domains

The maximum number of OMNIDEX domains per database is 199. This figure is for a database where all OMNIDEX masters use an integer (type I, J, or K) search item. OMNIDEX domains where the master's SI is a character data type (U or X) require an additional index and may reduce this maximum number.



## **Number of keyword keys**

- ❑ The maximum number of composite keyword keys per database is 200.
- ❑ There is no known limit to the number of fields that you can directly install as keyword keys.

## **Number of excluded words**

There is no known limit to the amount of keywords that you can exclude from indexing for a given database. Over 3000 excluded words were successfully specified for databases during product testing.

## **Number of tables (data sets) that contain sorted keys**

The maximum number of tables that can contain sorted keys is 199, which is the maximum number of tables that can be defined for a database in TurboIMAGE.

## **Number of sorted keys**

- ❑ The maximum number of composite sorted keys that you can install on a database is 200.
- ❑ The number of fields that you can directly install as sorted keys is 255, assuming that no keyword keys are installed on the database.

## Table or domain maximums

A domain can consist of a master table and any detail sets linked to it during installation, a single master table, or a single, unlinked detail table.

### Entries per table and OMNIDEX ID value

The highest integer value that can be assigned as an OMNIDEX ID, and therefore the maximum number of entries per master table, is 2,147,483,647.

### Number of keyword keys

- ❑ The maximum number of composite keyword keys per domain is 200.
- ❑ There is no known limit to the number of fields in a table that you can directly install as keyword keys.

### Number of keyword key groups

Up to 63 keyword key groups can be defined per domain.

### Number of sorted keys

- ❑ The maximum number of composite sorted keys that you can install on a table is 200.
- ❑ The number of fields in a table that you can directly install as sorted keys is 255, assuming that no keyword keys are installed on the database.



## Key maximums

### Multiple keys

- ❑ The maximum number of characters that are indexed per keyword (the keyword length) is 32. Keywords (including non-parsed key values) longer than 32 characters are truncated to 32 characters in the indexes.
- ❑ The maximum length of a composite keyword key is 250 bytes.
- ❑ The maximum number of components in a composite keyword key is a function of the maximum length (250 bytes).
- ❑ There is no known limit to the number of keyword keys that can be grouped together. The largest group tested contained 455 keyword keys.
- ❑ There is no known limit to the number of keywords that can be indexed for a keyword key.

### Sorted keys

- ❑ The maximum internal length for a composite sorted key is 250 bytes. Note that internal length of a composite key is its length added to the length of the search item, when installed on a master table, or added to 4 (the length of the relative record number), when installed on a detail table.
- ❑ The maximum number of components in a composite sorted key is a function of the maximum length (250 bytes).

---

# Glossary

---

|                                |                                                                                                                                                                                                                                                     |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Ad hoc</i>                  | Suited for a particular purpose. Refers to searches that involve several arguments, usually for the purpose of determining trends and relationships (like the number of sales made by a particular sales person during a particular sales quarter). |
| <i>Argument</i>                | A value used to search or retrieve records.                                                                                                                                                                                                         |
| <i>Back out feature</i>        | The ability to undo a keyword search by calling DBFIND mode 13 in the Standard TurboIMAGE/iX interface, or by passing <; through the <i>keywords</i> parameter of an ODXFIND.                                                                       |
| <i>Boolean operators</i>       | AND ( , ), OR ( + ), and NOT ( , - ) operators used to combine keywords during an OMNIDEX retrieval.                                                                                                                                                |
| <i>Call Conversion library</i> | Procedures that automatically intercept calls to TurboIMAGE intrinsics and convert them into calls to OMNIDEX intrinsics. See the “Updating OMNIDEX Data” section of the “Topics” chapter, and Appendix C.                                          |
| <i>Calling errors</i>          | Syntactical errors that 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 a sorted key would yield a calling error.                                        |
| <i>Compatibility Mode</i>      | A term attributed to programs and library routines designed to run on Classic HP computers. They must be emulated as Native Mode when run on series 900 HPPA computers.                                                                             |
| <i>Compound item</i>           | An TurboIMAGE item comprised of more than one element (like a 5X50 type field).                                                                                                                                                                     |

|                               |                                                                                                                                                                                                                             |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Composite keys</i>         | A logical keyword or sorted key comprised several fields or parts of fields.                                                                                                                                                |
| <i>DBMGR</i>                  | The DISC database management utility. See “DBMGR”, on page 3-34, for more information.                                                                                                                                      |
| <i>Data type discrepancy</i>  | When the data stored in a particular field does not match the TurboIMAGE data type defined for that field.                                                                                                                  |
| <i>Detail chain</i>           | A group of detail records associated with a single TurboIMAGE master record.                                                                                                                                                |
| <i>Deferred updates</i>       | When updating the indexes for keyword keys, the update is not performed as data is updated in TurboIMAGE files, but is deferred for later updating.                                                                         |
| <i>Discrete Mode</i>          | The term applied to high-performance retrievals and updates that affect the key value only. Discrete mode IMSAM retrievals return only the key values stored in the indexes without returning a record.                     |
| <i>Domain</i>                 | An association of one or more data sets that are linked by a common unique identifier. See <i>SI domain</i> and <i>DR domain</i> .                                                                                          |
| <i>DR domain</i>              | An OMNIDEX domain (see <i>Domain</i> ) that consists of one detail data set that has been installed with Detail Record (DR) indexing.                                                                                       |
| <i>Exceptional conditions</i> | Errors that are incurred after an intrinsic has been called and prevent the call from executing successfully. Trying to read in a sorted key sequence beyond the last key value would yield an exceptional condition error. |
| <i>Excluded words file</i>    | See <i>Excluded words list</i>                                                                                                                                                                                              |
| <i>Excluded words</i>         | Words in a file loaded through OmniUtil to be excluded from the OMNIDEX keyword indexes.                                                                                                                                    |
| <i>Excluded words list</i>    | An ASCII file that contains all of the excluded words. See “Excluding Words from Indexing”, on page 2-22 of the “OMNIDEX Installation” chapter.                                                                             |
| <i>General intrinsics</i>     | OMNIDEX intrinsics that are used to perform locks, updates, and other general functions.                                                                                                                                    |

|                                            |                                                                                                                                                                                                                                                                                |
|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Generic search or Generic retrieval</i> | An OMNIDEX search using a partially specified argument, or a keyword argument that ends in an at sign ( @ ).                                                                                                                                                                   |
| <i>Grouping</i>                            | A way to treat several keyword keys as one logical entity during keyword retrieval. See “Grouping”, on page 2-9 of the “OMNIDEX Installation” chapter, for more information.                                                                                                   |
| <i>IMSAM</i>                               | The <b>IMAGE Sequential Access Method</b> , which provides sorted-sequential and partial-key access on IMAGE databases. IMSAM is used to refer to the intrinsics that are called to perform sorted key retrievals, as well as the intrinsics used to maintain OMNIDEX indexes. |
| <i>Intrinsic</i>                           | a predefined operation that can be called from a program. See the “Intrinsics” chapter of the <i>OMNIDEX ImagePlus SDK API Guide</i> for more information.                                                                                                                     |
| <i>Key</i>                                 | A field that is used to select records from a data set. In OMNIDEX, this includes keyword and sorted keys as well as TurboIMAGE keys.                                                                                                                                          |
| <i>Key options</i>                         | Enhancements to OMNIDEX keys that enable certain retrieval features (like Soundex) or update features (like Batch Indexing). See “Adding key options”, on page 2-7, for more information.                                                                                      |
| <i>Keyed retrieval</i>                     | Retrieval by an OMNIDEX or TurboIMAGE key.                                                                                                                                                                                                                                     |
| <i>Keyword</i>                             | A word or number that is indexed in a keyword key and stored in the OMNIDEX indexes.                                                                                                                                                                                           |
| <i>Keyword key</i>                         | A type of OMNIDEX key that is used to find records by a keyword argument or combination of keywords arguments. See “Determining Retrieval Requirements” in the “Topics” chapter for more information.                                                                          |
| <i>Keyword list</i>                        | The list of arguments and operations passed through the keywords parameter in an ODXFIND keyword search.                                                                                                                                                                       |
| <i>Keyword retrieval</i>                   | Refers to retrieval of record identifiers by any keyword value or combination of keyword values against a keyword key.                                                                                                                                                         |

|                                |                                                                                                                                                                                                                                                                                                       |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>List compression</i>        | Converting the results of a keyword search from individual detail records, to record complexes. This is accomplished by calling ODXFIND mode 30 after a keyword search on a linked detail.                                                                                                            |
| <i>Mode</i>                    | A parameter of an intrinsic that specifies what type of action is to be performed.                                                                                                                                                                                                                    |
| <i>Multifind</i>               | A retrieval feature that supports OMNIDEX retrievals using field values from previously qualified records as keyword arguments for keyword searches across domains.                                                                                                                                   |
| <i>Native mode</i>             | A term attributed to programs and library routines that can be directly executed on a Series 9nn HPPA computer.                                                                                                                                                                                       |
| <i>Native Language Support</i> | The collating of 8-bit extended character sets (Arabic8, Greek8, Kana8, Roman8, Turkish8) to ASCII as handled by OMNIDEX and the HP 3000.                                                                                                                                                             |
| <i>Noise words</i>             | Words that occur frequently or are useless for retrieval (like "the" or "therefore").                                                                                                                                                                                                                 |
| <i>OMNIDEX detail</i>          | A detail set linked to an OMNIDEX master during installation.                                                                                                                                                                                                                                         |
| <i>OMNIDEX domain</i>          | See <i>Domain</i>                                                                                                                                                                                                                                                                                     |
| <i>OMNIDEX ID</i>              | A binary integer (I2, J2, or K2) value that is used to identify a DR detail set, or an OMNIDEX master set and any associated details. Also used as a pointer when indexing the keywords for a domain. It can be an I2, J2, or K2 TurboIMAGE search item, or a value that is maintained in background. |
| <i>OMNIDEX indexes</i>         | Files containing the key or keyword values and record identifiers that are used by OMNIDEX during retrievals.                                                                                                                                                                                         |
| <i>OMNIDEX master</i>          | A master data set that contains one or more keyword keys, or a master that is linked to one or more details that contain keyword keys.                                                                                                                                                                |
| <i>OMNIDEX SI</i>              | The TurboIMAGE search item field in an OMNIDEX master. It may also be the OMNIDEX ID for the set.                                                                                                                                                                                                     |



|                                 |                                                                                                                                                                                                                                                        |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>One-to-many relationship</i> | Refers to data entities where one entity corresponds to many other entities, as in the case of a search item value to many detail records.                                                                                                             |
| <i>One-to-one relationship</i>  | Refers to data entities where one entity corresponds only to one other entity, as in the case of a search item value to a master record.                                                                                                               |
| <i>Operator</i>                 | Any special character or token used in OMNIDEX retrievals. For example, the samelist operator ( * ).                                                                                                                                                   |
| <i>Parsing</i>                  | Separating a character string into individual keywords.                                                                                                                                                                                                |
| <i>Partial-key retrieval</i>    | Refers to a sorted key retrieval where a partially specified search argument is used to retrieve one or more records.                                                                                                                                  |
| <i>Privileged mode</i>          | A special mode in the HP operating system that bypasses normal operating system checks. DBINSTAL uses privileged mode. OMNIDEX intrinsics do not use privileged mode.                                                                                  |
| <i>Qualifying count</i>         | The number of search items or TurboIMAGE record numbers that qualify in an OMNIDEX keyword retrieval.                                                                                                                                                  |
| <i>Record complex</i>           | An affiliation of records in an SI domain that all contain the same OMNIDEX SI value. Also refers to keys that qualify record complexes (keyword keys in master data sets, and keyword keys installed with the Record Complex option).                 |
| <i>Record specific</i>          | The default indexing used for keyword keys installed on details, which supports the retrieval of record complexes by SI values, or individual detail records by relative record number. Also refers to keys that can locate individual detail records. |
| <i>Record number</i>            | An internally maintained value associated with the position in a detail file of individual detail records.                                                                                                                                             |
| <i>Relational item</i>          | The key used in a Multifind operation that establishes a relation between the two sets involved. The key for which an &, or an &filename, is used as a search argument.                                                                                |
| <i>Relational operator</i>      | The greater than (>), greater than or equal to (>=), less than (<), less than or equal to (<=) and equals (=) operators used to specify a relational retrieval.                                                                                        |

|                                 |                                                                                                                                                              |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Relative record number</i>   | See <i>Record number</i>                                                                                                                                     |
| <i>Retrieval intrinsics</i>     | Intrinsics that are used to search for or retrieve records.                                                                                                  |
| <i>Root file</i>                | A file that contains information about the database structure.                                                                                               |
| <i>SI domain</i>                | An affiliation of one master set and one or more detail sets, linked at installation, whose keyword values are all indexed in the same indexes.              |
| <i>Samelist operator</i>        | The asterisk ( * ), which tells OMNIDEX that you wish to further qualify the list of IDs.                                                                    |
| <i>Search item (SI)</i>         | An TurboIMAGE field within a data set that is used for calculated access in master sets and chained access in detail sets.                                   |
| <i>Segmented library (SL)</i>   | A collection of procedure segments. In version 3.0 of OMNIDEX, the OMNIDEX Intrinsic Switch Stub segments, are located in a segmented library (SL.PUB.DISC). |
| <i>Simultaneous path access</i> | The ability of OMNIDEX to search several keyword keys at once. Also known as multiple key access.                                                            |
| <i>Special character</i>        | A character that is parsed by OMNIDEX before being indexed. Likewise, a character that is interpreted by OMNIDEX to have a special meaning.                  |
| <i>Sorted key</i>               | A field or composite key specified for sorted access. Sorted key retrievals return records sorted by key values.                                             |
| <i>Sorted access</i>            | The ability to retrieve records in sorted order through the use of a sorted key.                                                                             |
| <i>Source</i>                   | In Multifind usage refers to the item, set or database that is supplying the data values used as search arguments.                                           |
| <i>Split retrieval</i>          | Refers to a keyword search on a record specific key that immediately precedes or immediately follows a search on a record complex key.                       |
| <i>Subfield</i>                 | In TurboIMAGE, the individual elements that comprise an array. For example, each X50 element is a subfield in a 5X50 array.                                  |

---

|                          |                                                                                                                                                                                                                                                                          |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Switch stub</i>       | A routine that allows Compatibility Mode programs to call OMNIDEX version 3.0 Native Mode intrinsics.                                                                                                                                                                    |
| <i>TPI-enabled</i>       | Refers to a database whose OMNIDEX indexes are accessible to the Standard TurboIMAGE/iX Interface to Third Party Indexing.                                                                                                                                               |
| <i>Table</i>             | See <i>Data set</i> .                                                                                                                                                                                                                                                    |
| <i>Target</i>            | When used in reference to Multifind, it refers to the domain, data set or field in which a list of keyword values is to be further qualified.<br>When used in reference to DISC utilities, it refers to a data entity on which a specified operation is to be performed. |
| <i>Textual data</i>      | Data containing words, phrases, and sentences (as opposed to numeric, or other fixed data).                                                                                                                                                                              |
| <i>Token</i>             | See <i>operator</i> .                                                                                                                                                                                                                                                    |
| <i>Translation table</i> | A text file that is loaded through OmniUtil to handle the translation of an 8-bit character set other than ROMAN8 by OMNIDEX's native language support capability.                                                                                                       |
| <i>USL</i>               | User-segmented library. It contains Relocatable Binary Modules that can be added to an SL.                                                                                                                                                                               |
| <i>Upshifting</i>        | Shifting of alphabetic characters to upper case in the OMNIDEX indexes.                                                                                                                                                                                                  |
| <i>Wild card</i>         | Any character that is used to imply a partial search argument. The at sign ( @ ) is used as a wild card in generic retrievals.                                                                                                                                           |



---

# Index

---

## Symbols

- ! Exclamation point
  - in keyword-only search B-6
  - in Soundex searches 2-13, 4-7, B-6
- " Quotes
  - in generic retrieval 4-9, B-7
  - parsing of B-7
- # Pound sign
  - as wild card B-6
  - as wildcard 4-7
- \$ Dollar sign
  - used in ID file searches 4-23, B-6
- & Ampersand
  - in Multifind 4-65, B-7
- () Parentheses
  - in Boolean expressions 4-14, B-7
- \* Asterisk
  - reloading ID list B-7
- + Plus sign
  - as OR operator B-7
- , Comma
  - as AND operator B-7
  - parsing of B-3
- Minus sign
  - as NOT operator B-7
- . Decimal points
  - parsing of B-4
- : Colon
  - as range operator B-7
- > Angle brackets
  - as relational operators B-7
- ? Question mark
  - as wild card B-7
  - as wildcard 4-7

- @ At-sign
  - as wildcard 4-7, B-6
- \ Backslash
  - in ASCII translation table 2-25

## Numerics

- 8-bit character sets
  - control codes 2-25
  - customizing translation 2-24
  - default translation B-8

## A

- Access
  - ad hoc searches 4-35
  - keyword vs. sorted 4-49
  - partial chained reads 4-47
  - serial reads 4-34
  - simultaneous keyed access 4-37
  - simultaneous multiple set 4-38, 4-57
- Ad hoc searches 4-35
- ALTPROG 3-58
- AM capability
  - DBMGR SCHEMA command 3-57
- Applications
  - on-line benchmarks 4-36
- Arguments
  - in Boolean operations 4-13
- Arrays and compound items A-3
  - binary 2-12
  - indexing as one field 2-8
  - indexing elements 4-40
- Automatic masters
  - eliminating 4-103

**B**

Batch Indexing (BI) option 4-80

    maintenance 4-80

    overview 2-14

    performance gains 4-53

Batch updates 4-52, 4-98

Binary data

    arrays 2-12

    No Parse option 2-11

    stored in ASCII fields A-5

Blanks and nulls, see Spaces and nulls 2-15

Blob Indexing option 2-8

Boolean operations 4-13

    on record specific keys 4-60

    operators 4-13

    order of precedence 4-14

    split retrievals 4-61

    with Samelist 4-4

Broken chains

    and DBMGR 3-53

    fixing 3-40

buffer size 4-99

**C**

Call Conversion library 1-16

CAP 3-58

Capacity changes

    data sets 3-45

    effect of TPI 4-93

    index files 4-88

CAPCHK 3-58

Character fields A-3

COBGEN utility 3-58

COBOL

    generating source code 3-58

Code fields

    and OMNIDEX 4-32

Commands

    DataView 3-21

    DBMGR 3-43

    OmniUtil 3-2

    retrieving records 1-14

Compatibility mode 1-17

Composite keys

    hierarchical 4-32

    integer components 2-5

    specifying components 2-4

    typecasting components 2-5

Composite keyword keys 4-44, 4-45

    applications 4-38

    combining fields 4-44

    data type 2-6, 4-44

    designing 4-44

    installing 2-4

    isolating parts of fields 4-40

    maximum length 4-44

    maximum number D-2

    rearranging fields 4-39, 4-45

    rules 4-44

Composite Sorted keys

    applications 4-39

Composite sorted keys

    combining fields 4-39, 4-47

    data type 2-7

    designing 4-46

    hierarchical 4-47

    Index-only retrievals 4-48

    installing 2-4

    isolating parts of fields 4-40

    rearranging fields 4-39

    restrictions 4-43, 4-46

    sort order 4-48

    sorted paths 4-48

    sorting on multiple fields 4-46

    supporting partial chained reads 4-47

Conventions

    used in this guide xi

**D****Data**

- binary stored in ASCII fields A-5
- converting to fit set structure 4-111
- copying to files 4-109
- effect on design 4-29
- loading detail sets 4-112
- loading from flat files 4-116
- ordered vs. free-form 4-29
- transferring 4-109

**Data retrieval**

- bearing on installation 4-41
- of key values only 4-42, 4-48
- qualifying count 4-35, 4-43
- sorting records 4-42

**Data sets**

- capacity and reloads 3-55
- changing capacity 3-45, 4-93
- in multilevel hierarchies 4-105
- inserting/deleting 4-73
- reindexing 3-12
- reloading 4-93

**Data types**

- ASK dates A-4
- binary stored in ASCII A-5
- character (U or X) A-3
- composite key components 2-5
- discrepancies A-4
- integer (I, J or K) A-1
- packed (P) fields A-2
- PowerHouse date A-4
- real (R) fields A-3
- supported by OMNIDEX A-1
- zoned (Z) fields A-2

**Database**

- adding many records 4-88
- changing structure 4-107
- converting from version 2.*nn* 2-13, 4-70
- copying 4-74
- creating 4-108
- DBMGR and security 3-48

- disabling TPI 4-54
- form sets listing 3-51
- generating a schema 3-56
- loading data from flat files 4-116
- logical design dilemmas 4-105
- maximums D-1
- modifying 4-102
- number of opens per process D-1
- reindexing 3-11
- removing OMNIDEX from 4-93
- renaming 4-75
- revising the schema 4-108
- streamlining 4-102
- structural information about 3-13

**Databases**

- releasing after DBMGR 3-47

**DATADEX**

- BUILD command 4-116
- program operation 4-110
- transferring data 4-109–4-110
- XFER command 4-110

**DataView**

- color configuration 3-19
- current table 3-22
- function keys 3-23
- getting help 3-23
- Main Menu 3-24
- menus and windows 3-23
- overview 1-14
- program operation 3-21
- running from OmniUtil 3-7

**Date fields**

- ASK dates 2-12, A-4
- keyword searches 4-10
- OMNIDEX access to 4-33
- PowerHouse dates 2-12, A-4

**DBFIND**

- mode 1 4-25
- TPI modes 4-25

**DBDELETE 4-98**

- DBINSTALL
    - enabling TPI 4-84, 4-86
    - removing OMNIDEX keys 4-94
  - DBIPUT 4-98
  - DBIUPDATE 4-98
  - DBLOAD 4-68, 4-78
    - loading data 4-109
  - DBMGR
    - and Third Party Indexing 4-81
    - benchmarks 3-35
    - capacity changes 4-98
    - CAPacity command 3-35, 3-45
    - changing database 3-45
    - commands 3-43
    - concurrent processes 3-39
    - COPY command 3-48
    - DELeTe command 3-49
    - ERAsE command 3-50
    - Exit command 3-51
    - FOrm command 3-51
    - Help command 3-52
    - PRImary command 3-52
    - QUIT command 3-53
    - RELoad command 3-53
    - reloads 4-98
    - REName command 3-55
    - SCHema command 3-56
    - security 3-37
    - Third Party Indexing 3-38
  - DBSCHEMA utility 4-108
  - DBUNLOAD 4-68, 4-78
  - DBUTIL utility 4-82
    - CREATE command 4-108
  - Detail data sets
    - broken chains 3-40
    - linking to masters 2-16, 4-56
    - list compression 4-23
    - minimum capacity 3-47
    - OMNIDEX 4-56
    - using IDADD 4-112
  - Detail Record (DR) indexing 4-62
    - DR domains 4-62
    - multiple set access 4-62
  - DISC\_XLNAME environment variable C-8
  - Disk space 4-96
    - determining availability 3-59
    - for keyword indexes 4-96
    - for sorted indexes 4-97
    - for unload files during indexing 4-97
  - Display
    - printing contents of 3-59
  - Domains 4-57
    - DR domain 4-57, 4-62
    - DR domain qualifying count 4-43
    - Grouping option 4-57
    - maximum number of sets permitted 4-57
    - SI domain 4-56, 4-57
    - SI domain qualifying count 4-43
    - simultaneous multiple set access 4-57
  - dset* Parameter
    - DBIGET vs. DBGET 4-21
- ## E
- Errors
    - recovering from indexing errors 4-79
  - Excluded word list 4-101
    - creating 2-22
  - Excluding keywords 2-22, 4-22, 4-42, B-2
    - see also No Exclude (NE) option
    - maximum number D-2
    - No Translate (NT) option 2-22
    - OmniUtil command 3-18
    - performance considerations 4-101
    - respecifying excluded words 4-79
  - Exclusive access 4-51, 4-53, 4-76



**F**

## Fields

- arrays A-3
- calculation fields 4-36
- character A-3
- code fields 4-32
- combining 4-44
- combining into keys 4-39
- data types A-1
- hierarchical 4-32
- indexing parts of 4-40, 4-45
- OMNIDEX design considerations 4-30
- rearranging 4-45
- selecting for OMNIDEX 4-37
- simultaneous keyed access 4-37
- sort paths 4-48
- sorting records by 4-47
- subitems 4-45
- textual and descriptive 4-31
- TurboIMAGE keys 4-31
- using in composite keys and fields 4-38

## Files

- Multifind 4-66

## FUTIL 3-58

**G**

## Generic arguments

- in keyword-only searches 4-22
- including special characters 4-9, B-7
- packed keys A-2
- ranges 4-13
- real keys A-3

## Grouping option

- and No Translate (NT) option 2-10
- and Record Complex (RC) option 2-10, 2-12
- and SI domains 4-57
- effect on SI domains 4-60
- integer fields A-1
- maximum number of groups D-3

overview 2-9

packed (P) fields A-2

real (R) fields A-3

restrictions 2-10

**H**

## Hierarchical fields

role in database design 4-32

## Hierarchical keys

composite 4-32

**I**

## ID Sort

*see* Keyword keys, sorting

IDADD utility 3-58, 4-111

## IMAGE

converting calls 1-17

DBGET intrinsic 4-48

field arrays A-3

## IMSAM

keyed access 1-16

## Indexes 1-12

batch updates 4-53, 4-98

changing capacities 4-91

checking capacities 4-89

deleting 4-73

disk space required 4-96

errors 4-79

file information 2-21

loading key values 2-28

logging updates to 4-85

maintaining capacities 4-88

maximum number D-1

releasing 4-88

user access to 4-88

version 3-15

**Indexing**

- after changing an installation 2-35
- after reloading sets 4-78
- automatic 4-51
- Batch Indexing (BI) option 4-80
- batch mode 4-80
- effect on ID files 4-23
- effect on index capacity 4-90
- installation changes 4-78
- jobs 4-80
- performance 4-99
- respecifying excluded words 4-79
- setting buffer size 2-30, 4-99
- TPI-enabled databases 2-29
- use of memory 4-101
- when is it necessary 4-78

**Index-only mode 4-48**

- retrievals 4-24, 4-42

**Installation**

- activating (creating indexes) 2-19
- activating a deferred installation 3-10
- Batch Indexing option 2-14
- changing 2-16, 2-34, 3-5, 4-72, 4-78
- composite keys 2-4
- correcting mistakes 2-16
- deferring activation 2-20
- finishing for a table (data set) 2-16
- Grouping option 2-10
- jobs 2-21, 4-76
- key options 2-7
- keying fields 2-3
- No Exclude option 2-15
- No Parse option 2-11
- prerequisites 2-2
- selecting tables (data sets) 2-3
- specifying key types 2-5
- Type Casting option 2-11
- Unique Key option 2-14
- verifying and testing 2-32
- version information 3-15

**Integer fields A-1**

- Grouping option A-1
- No Parse option 2-11

**Intrinsic switch stubs 1-17****Intrinsics**

- explicit calls 4-52
- implicit calls 4-52
- OMNIDEX 1-15

**Intrinsics library**

- redirecting calls to C-8

**Items**

- inserting/deleting 4-73

**J****Jobs**

- advantages in indexing 4-80
- DBMGR 3-41
- enabling/disabling TPI 4-81, 4-84, 4-86
- index loading 2-30, 4-80
- installation 2-21, 3-10, 4-76

**K****Key options**

- changing 2-18
- installing 2-7
- keyword keys 2-8
- menu 2-7
- sorted 2-14

**Keys**

- changing keytype 2-18
- changing options 2-18
- combining fields 4-39
- data accessed 4-29
- hierarchical 4-47
- keyword 1-2
- modifying 4-72
- on arrays 4-40
- removing 3-10
- sorted 1-6
- verifying presence 2-32

- Keyword access
  - compared to sorted 4-49
  - qualifying count 4-43
  - vs. serial reads 4-34
- Keyword keys 1-2
  - binary arrays 2-12
  - indexing nulls 2-15
  - maximum length D-4
  - maximum number D-2, D-3
  - on search items 2-11
  - options 2-8, 2-14
  - searching several 4-3
  - sorting 2-31
- Keyword search
  - see also Split retrieval
  - across databases 4-22
  - across keys 4-3
  - across tables (sets) 4-5
  - argument parsing B-5
  - back-out feature 4-3
  - batch mode 4-6
  - Boolean operations 4-13
  - compared to serial reads 4-34
  - date fields 4-10
  - date storage formats 4-10
  - general discussion 1-3
  - generic 4-8
  - multiple-field 4-37
  - multiple-keyword 4-37
  - no records qualify 4-6
  - operations supported 4-12
  - pattern-matched 4-7
  - performance considerations 4-34
  - ranges 4-12
  - relational operations 4-15
  - Soundex 4-7
  - transferring data from 4-22
- Keyword-only search 4-42
  - ODXFINd modes 4-22
  - uses 4-22
- Keywords
  - determining number 4-100
- KSAM
  - eliminating 4-104
- L**
- Limits D-1
  - number of records 3-47
- List compression 4-23
  - and split retrievals 4-61
  - defined Glos-4
- M**
- Maintenance 4-68
  - installation changes 4-78
  - reinstallation required 4-69
  - reloading detail sets 4-78
  - reloading indexes 4-78
  - respecifying excluded words 4-79
- Master data sets
  - correspondence to details 4-56
  - hashing 3-47, 4-34
  - minimum capacity 3-47
  - OMNIDEX 4-56
- Maximums D-1
- Mode options 4-21
- Multifind 4-63
  - from memory 4-65
  - meaningful correspondence 4-64
  - qualifying count 4-66
  - requirements 4-64
  - role in design 4-105
  - role of search items 4-63
  - supporting 4-31
- Multiple keys
  - excluded words B-2
  - parsing indexed data B-1
  - spaces and nulls B-2
  - special characters B-2
- Multi-set searches 4-5
  - effect of linking tables 4-58

**N**

- No Exclude (NE) option 2-15
  - see also Excluding keywords
- No Parse (NP) option 2-10
- No Translate (NT) option 2-15
  - and Grouping 2-10
  - effect on excluded words 2-22

**O**

- ODXM210 3-58, C-2
- ODXVER 3-58
- OMNIDEX
  - access capabilities 4-36
  - application development 1-10
  - changing installation 2-16, 4-72
  - components 1-13
  - configuring 3-17
  - data retrieved 4-41
  - detail sets 4-56
  - domain 4-57
  - general intrinsics 1-16
  - implementation 1-12
  - intrinsic libraries 1-15
  - master sets 4-56
  - overview of software 1-2
  - performance 4-96
  - principal features 1-8
  - reinstallation 4-69
  - removing 4-93
  - retrieval options 1-9
  - retrieval speed 1-8
  - retrieving keywords only 4-22, 4-42
  - technical support xiii
  - upgrading from earlier version 4-70
  - utilities 1-13
  - what keys are installed 3-9, 3-14
- OMNIDEX design
  - code fields 4-32
  - composite keys 4-43
  - date fields 4-33
  - field considerations 4-30

- hierarchical fields 4-32
- Multifind 4-105
- report performance 4-33
- sort paths 4-48
- text and description fields 4-31
- OMNIDEX domain
  - and record complexes 4-58
  - DR details 4-62
  - ID files 4-23
  - maximum number D-1
  - searching across 4-63
  - SI domains 4-57
- OMNIDEX IDs
  - adding ID values 4-111
- OMNIDEX root file 2-21
  - changes affecting 4-73
  - corrupted 4-72
- OmniUtil
  - configuring 3-18
  - Database Options Menu 3-17
  - exiting 3-8
  - function keys 3-4
  - getting help 3-4
  - Index Installation and Maintenance Menu 3-8
  - Main menu 3-6
  - menus 3-3, 3-6
  - overview 1-13
  - program operation 2-2
  - screen colors/attributes 3-19, 3-20
  - Show Information Options Menu 3-13
  - using the program 3-2
  - version information 3-8
  - viewing progress 3-5

**P**

- Packed fields A-2
- Parameters
  - differences from IMAGE 4-19

---

**Parsing**

- # \$ % & B-3
- binary Sorted keys B-5
- Blob option 2-9
- commas ( , ) B-3
- compound items 2-8
- decimal points ( . ) B-4
- disabling 2-11
- excluded words B-2
- Multiple keys B-1
- Sorted Keys B-5
- spaces and nulls B-2, B-5
- special characters B-2

**Passive Error Handling**

- recovering from indexing errors 4-79

**Paths**

- in OMNIDEX 4-56
- simultaneous access 4-37
- sorted 4-48

**Pattern matching 4-7, 4-26****PCLINK 3-59****Performance**

- indexing 4-99
- update 4-50

**Privileged Mode (PM) 4-70**

- during installation 2-2
- index population 4-80

**Product support xiii****Programs**

- vendor-supplied 4-52

**PSCREEN 3-59****Q****Qualifying count 4-43**

- keyword-only search 4-22
- Multifind 4-66
- record compaction 4-61

**Query**

- SAVE command 4-109

**R****Range operations**

- ASCII keys 4-27
- binary keys 4-27
- generic 4-13
- in keyword-only searches 4-22
- integer keys A-1
- keyword keys 4-12
- packed keys A-2
- real keys A-3
- tokens supported 4-12
- zoned keys A-2

**Real fields A-3****Real time indexing**

- enabling TPI 4-51
- performance 4-52
- through Call Conversion 4-51

**Record Complex (RC) option**

- installing 2-12

**Record complexes 4-58**

- converting from individual detail records 4-23

**Record specific keys 4-57**

- list compression 4-23
- split retrievals 4-60

**Records**

- Multifind correspondence 4-64
- qualifying count 4-35, 4-43
- retrieving 4-41
- sorting 4-42, 4-47

**Recovery from catastrophe 4-87****REGTEST 3-59****Reinstallation 4-69**

- adding keys 2-16
- batch mode 4-76
- changing key options 2-18
- changing key types 2-18
- changing keys 2-34, 4-72
- corrupt OMNIDEX root file 4-72
- removing keys 2-17
- when is it necessary 4-69

- Relational operations 4-26
  - keyword keys 4-15
- Releasing databases 4-88
- Reloads 4-93
  - onto a different device 3-53
- Removing OMNIDEX keys 4-93
  - from many databases 4-95
- Reports 4-33
  - conditional search statements 4-35
  - converting to on-line 4-35
  - data sorts 4-36
  - record counts 4-35
  - totaling calculation fields 4-36
- Retrievals
  - see also Keyword search
  - see also Split retrieval
  - after keyword search 4-6
  - batch mode 4-6
  - Index-only mode 4-48
  - key values only 4-42, 4-48
  - of records 4-41
  - OMNIDEX performance 4-41
  - partial-key 4-32

## S

- Search items
  - adding an integer SI 4-104
  - integer 4-102
  - No Parse option 2-11
  - qualifying count 4-43
  - role in Multifind 4-63
  - use in composite keys 4-47
- Searches
  - see also Split retrieval
  - across data sets 4-38, 4-57
  - ad hoc requests 4-35
  - multiple-field 4-37
  - multiple-keyword 4-37
  - OMNIDEX possibilities 4-36
  - record specific and record complex 4-61

- Security
  - when generating schemas 3-57
- Serial reads
  - vs. OMNIDEX retrievals 4-34
- SET TPI NOXM 4-86
- SHOWDISC 3-59
- SI/ID
  - adding to data set 4-111
- SLs
  - copying segments across 3-59
- Sort paths
  - eliminating 4-102, 4-103
  - role in database design 4-48
- SORT utility 4-110
- Sorted access 1-7
  - across multiple fields 4-47
  - by several fields 4-42
  - compared to keyword 4-49
  - in reports 4-36
  - integer keys A-1
  - packed keys A-2
  - qualifying counts 4-26
  - real keys A-3
  - retrieving key values only 4-24, 4-42
  - retrieving records 4-25, 4-42
  - sort paths 4-48
  - through DBFIND 4-25
  - zoned keys A-2
- Sorted keys 1-6
  - changing installation 4-72
  - indexing blanks and nulls 2-15
  - maximum length D-4
  - maximum number D-2, D-3
  - options 2-14
  - packed type arguments A-2
  - real type arguments A-3
  - reindexing 3-12
  - reinstallation 4-69
- SOS 3-59
- Soundex arguments 4-7
- Soundex option
  - and Grouping 2-13, 4-7

- installing 2-13
- Spaces and nulls B-2
  - indexing 2-15
  - parsing B-5
- Special characters
  - in 8-bit character sets 2-25
- Split retrieval 4-60
  - defined Glos-6
  - supporting 4-61
- SQUISHER 3-59
- Status array
  - OMNIDEX intrinsics 4-21
- System failure
  - recovering from 4-87

**T**

- Tables
  - maximum number of entries D-3
- Technical support xiii
- TPI
  - disabling 4-54
  - effect on transaction management 4-85
  - enabling 4-83
  - enabling/disabling 4-81
  - indexing operations 2-29
- TPIDRVR 3-59
- TPIXLCFG 3-59
- Transaction Manager
  - logging indexing 4-85
- TurboIMAGE/iX Interface 1-10
  - see also TPI
  - access to XLOMNIDX C-2
  - disabling index updates 4-54
  - disabling TPI 4-54
  - effect on database utilities 4-81
  - enabling for many databases 4-84, 4-86
  - enabling/disabling TPI 4-81
  - recovery from catastrophe 4-87
  - releasing indexes 4-88
  - seeing if its enabled 4-81

- Type Casting option
  - binary arrays 2-12
  - overview 2-11

## U

- Unique Key (UK) option
  - overview 2-14
- Updates
  - eliminating indexing overhead 2-14
  - IMAGE-only 4-53, 4-80, 4-98
  - improving performance 4-52, 4-98
- Upgrading OMNIDEX 4-70
- Utilities

- see DataView
- see DBMGR
- see OmniUtil
- ALTPROG 3-58
- CAP 3-58
- CAPCHK 3-58
- COBGEN 3-58
- FUTIL 3-58
- IDADD 3-58
- ODXVER 3-58
- PCLINK 3-59
- PSCREEN 3-59
- REGTEST 3-59
- SHOWDISC 3-59
- SOS 3-59
- SQUISHER 3-59
- TPIDRVR 3-59
- TPIXLXFG 3-59
- XM 3-59

## V

- Vendor-supplied programs 4-52
- Version information
  - for OMNIDEX installation 3-15

## W

---

## W

Wildcard tokens 4-7

## X

XLOMNIDX

checking the version of 3-58

XLSWITCH 3-59, C-2

XM 3-59

## Z

Zoned fields A-2

and No Parse option 2-11



