

SUPERDEX

User Manual **Version 4.2**

All updates to or derivatives of the SUPERDEXTM computer software provided herein are copyrighted and may not be copied except for archive purposes, to replace a defective copy, or for program error verification by Licensee. Copyrighted material may not be copied onto any media (e.g. magnetic tape, paper tape, disc memory cartridges, read-only memory, etc.) for any other purposes. The authorization to duplicate copyrighted materials hereunder shall not be construed to grant the Licensee or Licensee's customer the right to use copyrighted SUPERDEX material in any manner other than which is provided in this agreement or otherwise approved in writing by Bradmark Technologies, Inc..

© 2000, 1988, 1993 Bradmark Technologies, Inc.

All rights reserved

Printed in the U.S.A. (October 2000)

AdvanceLink, Business Basic, Business Report Writer, HP, IMAGE, TRANSACT, TurboIMAGE, and TurboIMAGE/XL are trademarks of Hewlett-Packard Company

ASK2 and VISIMAGE are trademarks of ARES

Business Session is a trademark of Tynlabs Corporation

dBASE, dBASE III, and dBASE III Plus are trademarks of Ashton-Tate Corporation

DBGENERAL is a trademark of Bradmark Technologies, Inc.

DIF is a registered trademark of Software Arts Products

ENQUIRE and SUPERDEX are trademarked product names of Bradmark Technologies, Inc. for the SI-IMAGE and ENQUIRE packages developed and implemented by Dr. Wolfgang Matt

FASTRAN is a trademark of Performance Software Group

Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation

Macintosh is a registered trademark of Apple Computer, Inc.

MPEX/3000 is a trademark of Vesoft Inc.

PowerHouse, QUIZ, QUICK, and QTP are registered trademarks of Cognos Incorporated

Reflection is a registered trademark of Walker Richer & Quinn, Inc.

SPEEDWARE is a trademark of Speedware Corporation

SYDAID is a trademark of Sydes

About this manual

In writing this manual, we have assumed that you have working knowledge, although not internal knowledge, of IMAGE and the HP3000.

All references to IMAGE in this manual and throughout the SUPERDEX package also apply to TurboIMAGE and TurboIMAGE/XL unless otherwise noted.

This manual is arranged in the following format:

Section 1 provides an *Overview* of the SUPERDEX package, its capabilities, and benefits. It also describes how SUPERDEX works and how it maintains compatibility with IMAGE and its facilities.

Section 2 overviews the various *Access methods* available in SUPERDEX for qualifying and retrieving entries in IMAGE databases, with one chapter per method.

Section 3 describes the procedures used in *Configuration/Establishing SI-indices* in SUPERDEX for use with your databases to provide quick retrieval of data entries.

Section 4 discusses the various methods utilized in *Programming* with SUPERDEX to add, update, delete, qualify, and retrieve entries in SUPERDEX, and gives examples of each.

Section 5 describes the *Intrinsics* provided with SUPERDEX as enhancements to the IMAGE intrinsics, as well as new SUPERDEX intrinsics.

Section 6 discusses the various *Maintenance* considerations for SUPERDEX'ed databases, as well as the use of various *Utilities* to access and maintain them.

Appendix A contains a copy of the Hewlett Packard Third-Party Indexing Interface External Specifications.

Appendix B examines various *Internal structures* used for SUPERDEX, including the method for calculating the capacity of the SI-dataset.

Appendix C documents *Maximum limits* that are imposed for SUPERDEX configuration.

Appendix D lists *Error and exceptional conditions* for SUPERDEX intrinsics, utilities, and programs with their meanings and recommended actions.

Finally, the **Index** is a complete index to the manual.

Table of Contents

Section 1: Overview	1-1
Enhancements and Compatibility	1-4
Compatibility	1-5
New Account Structure	1-6
Database Availability	1-7
SuperSELECT Now has WINDOWS!	1-7
New SIPATH with WINDOWS!	1-7
ASK Date Index Option	1-7
PowerHouse Date Index Option	1-8
15 Custom Separators for Keywords	1-8
Functionality	1-9
Sample applications	1-9
Multiple keys in master and detail datasets	1-10
Concatenated keys containing multiple fields	1-10
Sorted sequential retrieval	1-10
Keyword retrieval	1-10
Byte Offsets and Lengths	1-10
Generic and partial key retrieval	1-11
Approximate match retrieval	1-11
Greater-than, less-than, and range retrieval	1-11
Grouped retrieval	1-11
Super-grouped retrieval	1-12
Relational access: multiple criteria retrieval	1-13
Relational access: multiple fields, sets, and bases	1-13
Custom indexing	1-13
Independent non-IMAGE indexing	1-13
Concepts	1-14
Terminology	1-17
Access principles	1-21
Adding, updating, and deleting entries	1-21
Qualifying and retrieving entries	1-21
Indexed access Vs relational access	1-21
Compatibility	1-22
Data types	1-22
Status array	1-22
Error and exceptional condition handling	1-22
Application programs	1-22
Fourth-generation languages	1-24
Native Language support	1-24
7-bit support for Swedish language	1-24

Section 2: Access methods 2-1

- Multiple keys in master and detail datasets 2-3
 - Functionality 2-3
 - Application 2-3
 - Index Substrings 2-3
 - Implementation 2-3
 - Operation 2-3
 - Efficiency 2-5
- Concatenated keys containing multiple fields 2-6
 - Functionality 2-6
 - Application 2-6
 - Index Substrings 2-6
 - Implementation 2-6
 - Operation 2-7
 - Efficiency 2-7
- Sorted sequential retrieval 2-8
 - Functionality 2-8
 - Application 2-8
 - Index Substrings 2-8
 - Operation 2-8
- Keyword retrieval 2-10
 - Functionality 2-10
 - Application 2-10
 - Implementation 2-10
 - Operation 2-12
 - Efficiency 2-12
 - Maintenance 2-14
- Generic and partial key retrieval 2-15
 - Functionality 2-15
 - Application 2-15
 - Index Substrings 2-15
 - Implementation 2-15
 - Operation 2-15
 - Efficiency 2-16
- Approximate match retrieval 2-17
 - Functionality 2-17
 - Application 2-17
 - Index Substrings 2-17
 - Implementation 2-17
 - Operation 2-17
- Greater-than, less-than, and range retrieval 2-20
 - Functionality 2-20
 - Application 2-20
 - Implementation 2-20
 - Operation 2-20
- Grouped retrieval 2-23
 - Functionality 2-23
 - Application 2-23
 - Implementation 2-23
 - Operation 2-24
- Super-grouped retrieval 2-25
 - Functionality 2-25
 - Application 2-25

Implementation	2-26
Operation.....	2-26
Maintenance	2-26
Relational access: multiple criteria retrieval.....	2-27
Functionality	2-27
Application.....	2-27
Implementation	2-27
Operation.....	2-27
Efficiency	2-28
Relational access: multiple fields, sets, and bases	2-29
Functionality	2-29
Application.....	2-29
Implementation	2-29
Operation.....	2-29
Custom indexing	2-31
Functionality	2-31
Application.....	2-31
Implementation	2-31
Operation.....	2-31
Independent indexing.....	2-32
Functionality	2-32
Application.....	2-32
Implementation	2-32
Operation.....	2-34
Maintenance.....	2-34
Section 3: Configuration / Establishing SI-indices	3-1
Installation.....	3-2
Enabling a Database for SUPERDEX 4.0/TPI	3-2
Disabling a Database for SUPERDEX 4.0/TPI	3-2
Configuration overview	3-4
Defining SI-keys and SI-paths	3-4
Naming SI-paths.....	3-4
Simple Vs concatenated SI-keys and their lengths	3-6
Compound items	3-6
Keyworded SI-paths.....	3-8
Grouped SI-paths	3-10
Super-grouped SI-paths.....	3-10
Custom SI-paths.....	3-11
Independent SI-paths.....	3-11
Blank SI-keys	3-11
Summary of restrictions	3-11
Excluding words from keywording.....	3-13
Definition and Purpose	3-13
Example.....	3-13
Default File	3-14
Customizing default characters.....	3-15
Configuring SUPERDEX using SIMAINT	3-21
Creating SI-item and SI-dataset(s).....	3-21
Operation.....	3-21
Access requirements	3-23
Input rules	3-23
Dialog phase.....	3-23

SIEXTLEN JCW for special concatenated SI-keys	3-25
Invoking SIMAINT	3-25
Invoking SIMAINT,SHARED	3-25
Defining database	3-26
Defining datasets	3-26
Defining associated SI-datasets	3-28
Bypassing datasets for independent SI-paths	3-28
Defining SI-paths and SI-keys	3-29
Defining simple SI-keys	3-29
Defining concatenated SI-keys	3-31
Defining keyworded SI-paths	3-33
Defining ASK Date SI-paths	3-34
Defining PowerHouse Date SI-paths.....	3-35
Defining grouped SI-paths.....	3-35
Defining super-grouped SI-paths.....	3-37
Defining SI-paths that are both keyworded and grouped	3-39
Defining custom SI-paths	3-39
Defining independent SI-paths	3-40
Defining keyword exclusion SI-path.....	3-42
Deferring indexing.....	3-42
Extension phase -- specifying SI-dataset(s) capacity	3-42
Indexing phase -- progress display	3-43
Example of configuring and establishing SI-indices for a database.....	3-45
Running SIMAINT in batch.....	3-47
Section 4: Programming	4-1
Adding, updating, and deleting entries.....	4-2
Adding and deleting entries with DBPUT and DBDELETE	4-2
Determining SI-key value	4-2
Custom SI-indices with SIUSER.....	4-2
Explicit SI-index management with DBPUTIX and DBDELIX.....	4-4
Independent SI-paths	4-4
Qualifying entries with DBFIND	4-5
Summary of DBFIND options.....	4-5
Indexed access Vs relational access	4-7
DBFIND modes	4-8
DBFIND arguments used for indexed access.....	4-8
DBFIND mode/argument examples	4-9
Finding entries using a partial key.....	4-13
Finding entries using a generic key	4-13
Finding entries greater than or equal to a specified value.....	4-15
Finding entries less-than or equal-to a specified value	4-16
Finding entries not equal to a specified value	4-16
Finding entries in a range of values.....	4-16
Finding entries in a concatenated SI-key.....	4-17
Finding entries in a group.....	4-17
Finding entries in a super-group.....	4-18
Finding entries in a compound IMAGE item	4-18
Finding entries by keyword	4-18
DBFIND arguments used for relational access	4-20
DBFIND mode/argument examples	4-23
Finding entries by ANDing multiple values.....	4-23
Finding entries by ORing multiple values.....	4-24

Finding entries by AND NOTing multiple values	4-24
Finding entries with combined Boolean operators	4-26
Active and backup SI-subsets	4-26
Successive refinement.....	4-28
Positioning on virtual SI-chain.....	4-29
Determining entry count of virtual SI-chain	4-29
Finding entries using multiple SI-paths in a dataset	4-31
Finding entries using multiple datasets.....	4-31
Finding corresponding entries in multiple datasets	4-34
Finding entries using multiple databases	4-34
Finding entries in multiple sets and bases using projection.....	4-36
Circumstances in which the SI-link must be specified	4-38
Qualifying entries in the active SI-Subset.....	4-38
Preparing the argument	4-39
High Speed OR'ing.....	4-39
Effect of DBFIND on the SI-pointer and current path.....	4-41
Retrieving entries with DBGET	4-43
DBGETs with Un-initialized SI-chain	4-43
Repositioning on an SI-chain	4-43
Reading SI-indices only	4-45
Reading multiple SI-indices with a single DBGET	4-45
Effect of DBGET on the SI-pointer and current path	4-46
Serially Reading All Entries	4-46
Additional programming considerations.....	4-48
Summary of effects of SI-intrinsics on the SI-pointer and current SI-path	4-48
Testing for the existence of SUPERDEX;	4-48
PowerHouse	4-50
TRANSACT.....	4-50
PROTOS	4-50
VISIMAGE	4-50
Native Language Support.....	4-51
Adding, updating, and indexing entries	4-51
Qualifying entries with DBFIND.....	4-51
Intrinsics.....	5-1
Enhancements	5-2
Summary of intrinsic enhancements	5-2
DBCONTROL intrinsic	5-3
Parameters	5-3
DBDELIX intrinsic	5-4
Parameters	5-4
Error handling	5-4
DBERASE intrinsic	5-6
Parameters	5-6
Improved speed in exclusive access mode	5-7
Recovery after abnormal abort.....	5-7
DBFIND intrinsic.....	5-8
Parameters	5-8
Effect of DBFIND on the SI-pointer and current path.....	5-17
DBGET intrinsic	5-18
Parameters	5-18
Effect of DBGET on the SI-pointer and current path	5-21
DBINFO intrinsic.....	5-22

Parameters	5-22
DBOPEN intrinsic	5-29
Parameters	5-29
DBPUTIX intrinsic.....	5-30
Parameters	5-30
Error handling.....	5-30
SITRANSLATE intrinsic	5-31
Parameters	5-31
Examples	5-32
SIUSER procedure.....	5-33
Parameters	5-34
DBPUT, DBUPDATE, and DBDELETE	5-34
SIMAINT utility.....	5-34
Maintenance and utilities.....	6-1
Database maintenance considerations	6-2
Error and Exceptional conditions	6-2
Database Maintenance Tasks	6-3
Redefining and reorganizing SI-paths	6-4
DBGENERAL interface	6-4
SIMAINT utility	6-5
Access requirements.....	6-5
Input rules	6-5
SIEXTLEN JCW for special concatenated SI-keys	6-6
Invoking SIMAINT	6-6
Specifying the database	6-6
Specifying datasets	6-6
Specifying SI-paths.....	6-8
Reorganizing SI-paths	6-8
Deleting SI-paths	6-9
DBLOAD Entry Point	6-10
LIST Entry Point	6-12
SCHEMA Entry Point	6-13
STRUCT Entry Point	6-16
Running SIMAINT in batch	6-16
SUPERDEX utility	6-18
Access requirements.....	6-18
Invoking SUPERDEX	6-18
Function Key Operation	6-19
Main Menu	6-19
Base Menu	6-21
Dataset Menu.....	6-22
Path Screen	6-24
Special Path Screen	6-26
Custom Path Screen.....	6-28
Path Display Screen.....	6-30
Item Screen	6-32
Item Definition Screen.....	6-34
Execute Menu	6-38
SIPATH utility.....	6-40
SITEST and SIREPAIR utilities.....	6-45
Access requirements.....	6-45
Invoking SITEST.....	6-45

Specifying the database.....	6-47
Specifying datasets.....	6-47
Specifying SI-paths	6-47
Specifying mode of operation.....	6-47
Mode 1 processing	6-48
Mode 2 processing	6-48
Running SITEST in batch.....	6-50
Invoking SIREPAIR.....	6-50
Specifying Input.....	6-50
Specifying request before update.....	6-51
SICOUNT utility.....	6-52
Invoking SICOUNT	6-52
Specifying the database.....	6-52
Specifying datasets.....	6-53
Specifying SI-paths	6-53
Process	6-53
SITRACE utility.....	6-56
Activating SITRACE	6-56
Function	6-56
Redirection of Output	6-56
SIDRIVER utility.....	6-58
Section 7: SuperSELECT.....	7-1
Invoking SuperSELECT	7-1
SuperSELECT - Method 1	7-1
SuperSELECT - Method 2.....	7-4
SuperSELECT - Method 3.....	7-5
SuperSELECT - Method 4.....	7-6
SuperSELECT - Windows	7-9
Appendix A: TurboIMAGE/iX Interface to Third Party Indexing Products:	A-1
Appendix B: Internal structures	B-1
SI-dataset structure.....	B-1
SI-dataset capacity	B-1
SI-item.....	B-3
SI-index	B-3
SI-pointer.....	B-4
SI-subset.....	B-4
Appendix C: Maximum limits.....	C-1
Appendix D: Error and exceptional conditions	D-1
SUPERDEX intrinsic error and exceptional conditions	D-2
SUPERDEX utility error and exceptional conditions.....	D-3
Program failures related to SUPERDEX	D-8
Supplement: TRANSACT interface	SUPP-1

SECTION 1:**Overview****Overview**

SUPERDEX is not a database management system, nor a programming language. It is a natural extension to IMAGE/SQL. SUPERDEX automatically creates and manages new B-tree indices in your databases and provides enhanced IMAGE/SQL-compatible intrinsics used by your programs automatically.

SUPERDEX has been designed and implemented to provide the most power and flexibility with the least amount of effort. SUPERDEX is so simple, it requires little training and takes only a few minutes to configure and minor program modifications to implement. Many SUPERDEX capabilities are accessible with no program modifications at all.

This section previews the SUPERDEX package, its capabilities, and its benefits.

Chapter 1 Why SUPERDEX?

Function explains the basic reason for SUPERDEX and it's capabilities.

Chapter 2 TPI Enhancements and Compatibility

Function is a description of the latest *enhancements* and *compatibility* issues in SUPERDEX Version 4.2 with reference to pre-TPI versions such as 3.1 or 3.2..

Chapter 3 Functionality

Function gives a brief description of the features that provide SUPERDEX's *Functionality* and a simple example of each.

Chapter 4 Concepts

Function explains the main *Concepts* of SUPERDEX and how they are used.

Chapter 5 Terminology

Function defines the *Terminology* used throughout SUPERDEX to identify its features and capabilities.

Chapter 6 Access Principles

Function overviews the *Access principles* within SUPERDEX for adding, updating, deleting, and retrieving entries.

Chapter 7 Compatibility

Function reviews *compatibility* issues, including compatibility with the IMAGE/SQL *status* array, the impact on existing application programs, and fourth-generation languages. Finally, Native Language Support and the Swedish versions are discussed.

CHAPTER 1:

Why SUPERDEX ?

Perhaps the best way to understand what SUPERDEX is all about is to understand why it was created, why it was implemented in the way it was, and why we believe you will find it to be a simple, straightforward method for achieving faster, more flexible access to your IMAGE, TurboIMAGE, TurboIMAGE/XL and IMAGE/SQL databases.

IMAGE/SQL, although a very functional and powerful database management system, lacks certain obviously-needed capabilities, such as:

- multiple keys in both master and detail datasets
- concatenated keys containing multiple fields
- sorted sequential retrieval
- byte offset keys with byte lengths
- automatic keywording and keyword retrieval
- generic and partial-key retrieval

- approximate match retrieval
- greater-than, less-than, and range retrieval
- grouping of functionally-equivalent fields
- relational access using multiple criteria
- relational access across multiple fields, datasets, and databases

SUPERDEX provides these desirable capabilities, and several others.

In IMAGE/SQL, for example, a master set indexes entries in a detail set, but there is no IMAGE/SQL structure that indexes a master set. To locate a master entry, you must specify the exact search field (key) value. To get around this limitation and permit entries to be located by multiple keys, many IMAGE/SQL databases are designed such that master-oriented information (entities such as customers, vendors, and parts) are placed in detail sets instead of master sets and indexed via automatic master sets. This leads to cumbersome, inefficient database structures, yet fundamental operations such as partial-key retrieval still cannot be performed without serially reading the dataset.

One fundamental requirement in designing SUPERDEX was to provide complete flexibility in searching for entries by allowing any field in any master or detail dataset to be used as a key. SUPERDEX permits you to designate any field--even every field--in master and detail datasets as a key.

Another requirement was to be able to specify a partial or generic key value in searching for entries. SUPERDEX can locate entries by a partial or generic key, multiple keys, or a range of keys. It can even find entries by any word contained in a key, which is called keyword retrieval.

SUPERDEX provides these capabilities for master and detail sets in the same way, so the same code that is written for detail datasets will work on master datasets, and vice-versa. In fact, ease of integration was one of SUPERDEX's primary requirements.

Another requirement was to minimize space, resources and impact on existing structures as little as possible. So, SUPERDEX requires only a single dataset with a single field in your database, in which it maintains special B-tree index structures. SUPERDEX manages these indices automatically from within the IMAGE/SQL intrinsics, to remain entirely within and compatible with IMAGE/SQL facilities such as transaction logging, unlike other indexing structures such as KSAM. These B-tree structures may be established, deleted, and reconstructed quickly on-the-fly, allowing very flexible indexing schemes like those inherent in relational databases.

Another IMAGE/SQL limitation is the inability to relate a master set to another master set. SUPERDEX permits logical master-master connectivity, and moreover, connectivity between multiple keys in a dataset, between any logically related datasets, and even across multiple databases.

The overall intent in creating SUPERDEX was to make it look and feel just like IMAGE/SQL--perhaps the next logical step for IMAGE/SQL--with fully-compatible enhanced intrinsics that provide additional functionality. In SUPERDEX, all access capabilities are available through DBFIND *mode 1* to find entries and DBGET *mode 5* and *6* to retrieve them. And, to make matters as simple and straightforward as possible, all selection criteria may be specified in the *argument* for DBFIND *mode 1*, permitting generic retrieval code to be written and the type of and scope of the retrieval specified by the user.

SUPERDEX's B-tree indices are maintained automatically by DBPUT, DBDELETE, and DBUPDATE and all status information is returned in the standard IMAGE/SQL *status* array.

And most important of all, SUPERDEX was designed to be the fastest method for retrieving entries in an IMAGE/SQL database. Lookups that would take minutes in IMAGE/SQL are done in seconds by SUPERDEX.

We know you will quickly see the areas in which SUPERDEX improves IMAGE/SQL, and appreciate the efforts that have been taken to make SUPERDEX as easy for you as its speed and power are to your users.

CHAPTER 2:

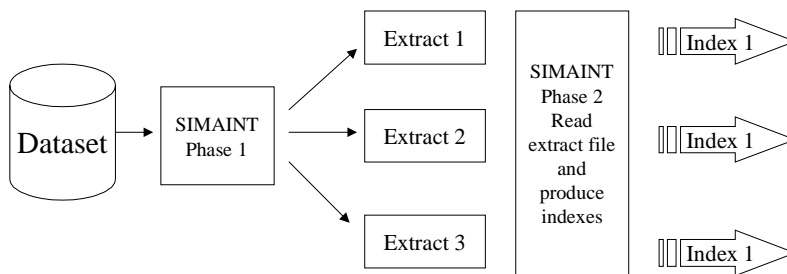
Enhancements and Compatibility

SUPERDEX Version four and later utilizes the IMAGE/SQL Third-Party Indexing (TPI) Interface. There are many enhancements and a few compatibility issues.

Enhancements

SIMAIN T Enhanced Speed on Index Creation

Performance of building indexes has been enhanced to read the dataset only once regardless of the number of indexes defined on the dataset. This may produce a substantial reduction in the time required to install or rebuild the indexes for datasets with multiple indexes. First, SIMAIN T reads the dataset once serially using a high-speed method and creates an extract file for each index defined. Then each extract file is taken in turn and processed to produce the index. Previously, the dataset would be read once for each index defined.



Testing shows a substantial portion of the time required to build an index is spent reading the dataset. Performance will vary, however the improvement can be estimated as:

$$\frac{\text{time_reading_dataset}}{\text{time_to_build_index}} * \frac{\text{number_of_indexes_in_dataset} - 1}{\text{number_of_indexes_in_dataset}}$$

Where:

time_reading_dataset

time_to_build_index

number_of_indexes_in_dataset

time for SIMAIN T to serially read the dataset.

complete time to read, sort and write single index.

total number of indexes defined on the dataset.

TPI Overhead Reduction

With TurboIMAGE C.07.##, TPI products have the ability to reduce overhead by signaling that there are not any TPI indexes involved with a particular dataset access. Previously, TurboIMAGE would execute the TPI code path in all cases. SUPERDEX version 4.2 has been enhanced to signal IMAGE and thereby reduce overhead for non-TPI enhanced dataset access.

- DBFIND and DBGET when the set does not have TPI and the trace facility, if applicable, is turned off.
- Serial DBGET (modes 2 and 3) for a set with TPI when the mode need not be promoted to other mode (possibly a chained mode) by the third-party software.
- Chained DBGET (modes 5 and 6) for a DBFIND of an IMAGE search item or a B-Tree item for a set with TPI.

SIQUIZ and SIQTP

The forth generation language front end programs SIQUIZ and SIQTP have been enhanced to invoke QUIZ and SIQTP via the currently set COGNOS UDC file. Previously, they ran QUIZ and QTP directly in a specified group and account, causing compatibility issues with newer versions of COGNOS products.

Compatibility

There are two areas of compatibility between SUPERDEX version 4 and earlier versions of SUPERDEX with which to be concerned.

The first is utilizing the two database options available in the SUPERDEX program or in SIMAINT. These two options are no longer necessary, and therefore, no longer supported. Databases that were flagged with option 2 (open the database twice) are modified automatically. Those which used option 3 (SUPERDEX indexes stored in a separate database) will need to move the indexes to the original user database. This can be accomplished by either redefining (and therefore reorganizing) the indexes or use DBGENERAL to add the new dataset(s) to the original database and copy the data from the SIBASE to the original database.

The second area of compatibility is existing code. In the TPI versions of SUPERDEX (4.0 and above) we attempted to support the previous modes as much as possible. Since the new modes were necessary in the TPI version there have been some changes.

In SUPERDEX Pre-TPI, locking of the user dataset(s) and SUPERDEX dataset(s) was very important. There were many different options for locking. One was for the programmer to explicitly lock the SUPERDEX dataset(s), while another was for SUPERDEX to lock ALL of the datasets, including the user's. All programs that utilized either of these two locking options MUST be modified. Programs that explicitly locked the SUPERDEX dataset(s) must either remove the calls to DBLOCK with the SUPERDEX dataset, or remove the SUPERDEX dataset(s) from the lock descriptor. Programs that had SUPERDEX implicitly lock all of the datasets, including the user's, must now call DBLOCK. They must explicitly lock the user dataset(s). The end result is that the program is responsible for locking the user dataset(s) and does not control the locking of the SUPERDEX dataset(s).

DBFIND mode 10 has been redefined. In pre-TPI versions of SUPERDEX DBFIND mode 10 worked as DBFIND mode 1, but did not return the number of records in the chain. In TPI versions of

SUPERDEX, DBFIND mode 10 forces a IMAGE/SQL DBFIND regardless of the set, item and argument parameters. DBFIND mode 21 is now used to set up the SUPERDEX chain, but not return the number of records.

The SUPERDEX DBFIND 2nn modes have also been redefined. The new 2nn modes support greater-than operations, similar to the 1nn greater-than-or-equal-to. The old 2nn modes supported a less-than-or-equal-to operation. These modes have been changed to 5nn.

The DBINFO modes have been redefined. The old 300 modes are not supported. There are several new DBINFO modes assigned to the 800 range. These modes should be used to retrieve the information in place of the old 300 modes.

❖ For TPI versions of SUPERDEX, we have provided a compatibility switch for those programs which utilized these modes. If the JCW SICOMPAT is not zero, ALL of the old SUPERDEX modes will be supported and none of the new TPI modes. This will allow programs to be modified as time allows so you do not have to modify every program before going to a TPI version of SUPERDEX.

New Account Structure

Since version TPI versions of SUPERDEX are tightly integrated with IMAGE/SQL, the SUPERDEX Indexing Engine (XL) is now located in the PUB.SYS account. The name of the engine is XLSUPRDX.PUB.SYS and must reside in the PUB.SYS account.

Along with the Indexing Engine, all of the maintenance and utility tools are located in the SYS account in a group called SUPERDEX.SYS. All of the programs (such as SUPERDEX, SIMAINT, SITEST, SIQUIZ, etc.) that were located in PUB.SUPERDEX are now in this new group. Many programs that were needed in pre-TPI (i.e. ALTPROG and SIBASE) have been removed.

The SUPERDEX account will still be created upon installation. The TPI demo will reside in the group DEMOTPI.SUPERDEX.

Some of these location changes were done out of necessity, while others for convenience. This structure will allow an Spectrum machine to run both SUPERDEX TPI and pre-TPI versions.

Database Availability

One of the most requested SUPERDEX enhancements was to add, delete, and maintain indexes in shared mode. Now SIMAINT can be run while other users are accessing the database, even if they are updating data. SIMAINT locks the appropriate dataset(s) during the organization process. This increases the availability of a database tremendously. Now, a path can be deleted, reorganized, and even created while users continue to access and maintain the database.

SuperSELECT Now has WINDOWS!

For SUPERDEX Level II customers, SuperSELECT now has a user-friendly windows interface. Using the script file, the database name, password, dataset name, and SUPERDEX path name can be passed to SuperSELECT windows. This allows the user to enter selection criteria through windows. The parameters are optional, which allows the user to even choose the dataset from which they wish to select.

In windows mode, SuperSELECT displays the number of records that qualify based on the selection(s) entered by the user. The user can enter multiple selection criteria, decreasing or increasing the number of qualified records. As each criteria is entered, SuperSELECT will display the number of qualified records so the user can be sure they have entered the correct selection. See the SuperSELECT section in the manual for complete details!

New SIPATH with WINDOWS!

SIPATH has been completely rewritten to utilize the windowing capability. The new window interface makes it very easy to request and display information about specific paths in the database.

See Section 6 for information on SIPATH, or just run it and press the 'HELP' function key!

ASK Date Index Option

SIMAINTE also supports a new index type, called the ASK Date format. SIMAINTE will allow the first subkey in an index to be defined as an ASK date item. It will then allow you to define the internal ASCII storage format (such as include the year and month only or include the full date: century, year, month and day), along with the ASK Date conversion format (ASK Pre-version 8.0 or ASK version 8.0 and later).

This will then store the index in an ASCII format, allowing full SUPERDEX retrieval on the ASK Date. It also stores the date in a format that the user can understand. So, programs that want to retrieve from an ASK Date item do not need to accept a date from the user and then convert it. The user simply enters a selection value (e.g. **9207@** or **9208@**).

PowerHouse Date Index Option

In addition to the ASK date format, SIMAINTE also supports the PowerHouse Date format. SIMAINTE will allow the first subkey in an index to be defined as a PowerHouse date item. It will then allow you to define the internal ASCII storage format (such as include the year and month only or include the full date: century, year, month and day).

This will then store the index in an ASCII format, allowing full SUPERDEX retrieval on the PowerHouse Date. It also stores the date in a format that the user can understand. So, programs that want to retrieve from a PowerHouse Date item do not need to accept a date from the user and then convert it. The user simply enters a selection value (e.g. **9207@** or **9208@**).

15 Custom Non-separators for Keywords

In version 4 the number of custom keyword non-separators has been increased from four (4) to fifteen (15). This allows much more power in customizing your database keywording. Now you can define up to 15 characters to use as non-separators for keywords.

CHAPTER 3:

Functionality

The various capabilities of SUPERDEX are covered here. More complete information about and examples of each feature appears in the *Access methods* section.

Sample applications

SUPERDEX may be used throughout your application systems in different ways to accomplish various operations. Some of the more common uses of SUPERDEX capabilities are listed here:

- **Customer lookup**
Customers stored in a master dataset need to be accessed by name, contact, phone number, and address. SUPERDEX could search on any field, keyword the contact so that either first or last name or both could be specified, and group together both lines of a two-line address so both are always searched. The customer name could be looked up by a partial or generic key.
- **Part lookup**
Users can enter partial part descriptions and the program retrieves all that qualify and displays them on the screen with their corresponding part numbers. SUPERDEX could treat the part description as a keyworded field, permitting any word or words within the description to be specified.
- **Part classification extract**
All part numbers start with a classifying character sequence, and it is necessary to retrieve all the parts which start with a certain sequence of characters, so those characters are specified as partial keys.
- **Mail room**
Everyone in a company has a mail-stop, but not all correspondence indicates it, so the mail clerks enter the addressee's name or partial address and get the mail stop for routing.
- **Text Management**
Comments and other text must be searchable by any word contained in an 80-character field; SUPERDEX could handle it as a keyworded field.
- **Library system**
Book titles, authors, and summary information stored in a master set and two related detail sets could be super-grouped together, permitting retrieval by any combination of criteria in a single operation.

Multiple keys in master and detail datasets

IMAGE/SQL lets you access a master set by only one field, and a detail set only via its related masters unless time-consuming serial reads are performed, forcing rigid applications and cumbersome database structures. Using simple techniques, SUPERDEX lets you access any dataset directly by any field, regardless of whether or not it is an IMAGE/SQL search field.

For example, a customer entry in a master dataset could be looked up by its customer number, customer name, contact name, or phone number.

Concatenated keys containing multiple fields

SUPERDEX permits multiple fields or truncated fields to be concatenated together and retrieval to be done on the entire concatenated value. This permits very specific lookups to be performed without having to read serially or down a chain to qualify entries that match on multiple fields because all fields may be contained in the key.

For example, a division number, group number, and partial account number could be concatenated together and looked up by the full combined value or any portion of the combined value.

Sorted sequential retrieval

IMAGE/SQL returns entries in chronological order, unless sorted paths are used. SUPERDEX returns entries in ascending or descending alphabetical order and, by using concatenated keys, provides more flexibility than sorted paths without the overhead.

For example, a classification number, account number, and date/time stamp could be concatenated together, and entries would be returned in chronological order within each account within each class.

Keyword retrieval

SUPERDEX lets you access entries by any word contained in designated fields. This technique is referred to as keywording.

For example, the entry "BRADMARK TECHNOLOGIES, INC." could be located by specifying either **BRADMARK**, **TECHNOLOGIES**, or **INC**.

Byte Offsets and Byte Length

SUPERDEX lets you define character indexes that include byte offsets, along with byte lengths. This provides the capability to build an index using only a portion of the IMAGE/SQL item. For example, the SUPERDEX index could start in the third position of an item for a length of five bytes.

Generic and partial key retrieval

IMAGE/SQL will not find an entry unless you specify its exact key value. SUPERDEX is far more forgiving: you may specify for any key or keyword a partial value or an embedded value with matchcodes.

For example, **GEN@** would find all the entries that begin with "GEN" and **MA??ER** would find all the entries that begin with "MA" followed by any two characters followed by "ER". **A#J3@** would find all the entries that begin with "A" followed by a single digit (0 - 9), then "J3" followed by any other characters.

Approximate match retrieval

IMAGE/SQL cannot find an entry that does not exist, but SUPERDEX can do the next best thing: find the nearest matching entry.

The alphabetic ordering of indices allows approximate match retrieval: if no matching entry exists, the nearest qualifying entry is returned, permitting a program to start reporting data at any alphabetic location.

Greater-than, less-than, and range retrieval

SUPERDEX is also capable of retrieving all entries that are:

- greater than or equal to a specified value
- less than or equal to a specified value
- not equal to a specified value
- within the range of two values

For example **>=1000** would find all the entries with amounts greater than or equal to 1000, **<=500** would find all entries with amounts less than or equal to 500, **<>10** would find all amounts not equal to 10, and **>=A@<=C@** would find all the entries that begin with the letters "A", "B", or "C".

Grouped retrieval

IMAGE/SQL can search only one field at a time. SUPERDEX lets you group multiple fields together at configuration time, and automatically searches them all at lookup time. IMAGE/SQL compound (arrayed) fields can be used as keys, and are grouped automatically.

For example, three fields containing phone numbers could be grouped together and would all be searched when retrieving by phone number.

Super-grouped retrieval

IMAGE/SQL can only search a single dataset at a time. SUPERDEX lets you form a super-group of a master set and one or more of its related (by IMAGE/SQL paths) detail sets and qualify master entries based on the contents of the related detail entries.

For example, a master set containing a book title related to a detail set containing authors and another detail set containing summary information could be super-grouped together, allowing master book entries to be qualified by title, author, and/or summary in a single operation.

Relational access: multiple criteria retrieval

Access may be performed using Boolean operations against multiple criteria, to retrieve:

- all entries that meet either criterion (**OR** operation)
- all entries that meet both criteria (**AND** operation)
- all entries that meet one criterion but not the other (**AND NOT** operation)

For example, find all the customers that have orders waiting to ship or on back-order; all customers who are more than 60 days delinquent and owe more than \$1000; all parts that are out of stock and not discontinued.

Relational access: multiple fields, sets, and bases

Relational queries may be performed based on multiple values across multiple fields, datasets, and databases using dynamically-joined indices. This provides the power of a relational database in accessing a regular IMAGE/SQL database.

For example, finding all the customers who have more than \$100,000 in annual activity, current orders pending, and who did that same amount of business last year requires access to the CUSTOMERS and ORDERS sets in the SALES database and the ORDER-SUMMARY set in the HIST database.

Custom indexing

SUPERDEX contains a facility for addressing non-standard indexing requirements for circumstances in which the index value cannot be determined automatically.

Examples of this are data type conversion, date reformatting, upshifting, key extraction, and stripping unneeded characters.

Independent non-IMAGE/SQL indexing

SUPERDEX is designed to index entries in IMAGE/SQL databases, but can also be used to index other types of files.

For example, separate word processing documents may be indexed by all the significant words in their document descriptions and accessed via their file names.

CHAPTER 4:

Concepts

SUPERDEX adds extended capabilities to IMAGE/SQL through the IMAGE/SQL Third-Party Indexing Interface.

These are the major concepts of SUPERDEX:

■ **B-tree indices instead of chains**

IMAGE/SQL uses doubly-linked lists to represent its chains. SUPERDEX uses *SUPERDEX indices* in B-trees which are contained in one or more standalone detail datasets in each database. These B-tree indices are automatically maintained and accessed by SUPERDEX intrinsics which are IMAGE/SQL-compatible. They are easy to configure and reconfigure.

■ **B-tree = automatic master set**

A SUPERDEX B-tree is functionally equivalent to an IMAGE/SQL automatic master set which provides access to a field in a dataset, commonly referred to as a "key". Like entries in an automatic master, SUPERDEX B-tree indices are added and deleted automatically. SUPERDEX easily replaces and enhances the functionality of automatic master sets with SUPERDEX indices.

■ **Master and detail sets treated equally**

In IMAGE/SQL, an automatic master may be related only to a detail set; in SUPERDEX, B-tree indices may be related to master sets as well as detail sets. In IMAGE/SQL, master and detail sets are handled differently: master sets are usually accessed via keyed reads (DBGET *mode 7*) and detail sets are accessed via DBFIND followed by DBGET *mode 5* or *6*. In SUPERDEX, both master and detail sets are accessed using a common method: DBFIND and DBGET *mode 5* or *6*, with DBFIND qualifying the entries and DBGET retrieving them--just like accessing an IMAGE/SQL path in a detail set. Of course, DBFIND and DBGET against IMAGE/SQL paths continue to function as in IMAGE/SQL.

■ **Entries returned in sorted order**

Entries are returned in ascending alphabetical order by SUPERDEX key value with DBGET *mode 5* and descending order with DBGET *mode 6*.

■ **Concatenated keys = sorted chains**

A SUPERDEX key may consist of multiple field values or substring field values concatenated together, permitting more flexible sorting than sorted chains and without the overhead.

■ **SUPERDEX indices self-maintaining**

SUPERDEX indices--like automatic master entries--are automatically added and deleted whenever DBPUT and DBDELETE are called. Additionally, DBUPDATE may also cause the indices to change automatically.

■ Explicit index maintenance possible

SUPERDEX indices may be added and deleted manually via new intrinsics. This permits custom indexing against IMAGE/SQL databases as well as indexing of external non-IMAGE/SQL files. This is called *Independent indexing*.

■ Zero or multiple indices per entry

A data entry may have zero or more SUPERDEX indices pointing to it, facilitating both multiple indexing (as used in keyword retrieval) and the exclusion of blank fields.

■ Improved handling of compound items

IMAGE/SQL does not allow compound items (which are also referred to as arrayed or repeating items) to be used as keys. In SUPERDEX, compound items may be used as keys and are handled such that every subitem in the item is automatically searched whenever the item is referenced.

■ Power in the DBFIND mode 1 argument

Most of SUPERDEX's powerful selection capabilities are available via DBFIND *mode 1*, with multiple values and operators included in the *argument* to define complex selection criteria. This permits generic code to be written and the user to specify the type and scope of retrieval.

■ Selection refinement and undo

SUPERDEX maintains the results of the current and previous DBFIND calls and manages them automatically; it also allows them to be manipulated explicitly. This permits successive DBFINDs to be used to refine and undo selections and to qualify entries across multiple fields, datasets, and databases.

■ Multiple relational syntax used

Boolean operations using multiple values can be specified in three common syntax.

SQL notation, as used with common SQL languages. The arguments can be entered with the words **AND**, **OR**, and **NOT**.

Infix notation, as used with common report-writers. The arguments can be entered with **+(and)**, **-(not)**, and **,(or)**.

Reverse Polish Notation (RPN), as used by HP calculators. In RPN, the operator follows the two values to which it applies.

CHAPTER 5:

Terminology

Several new terms are used by SUPERDEX to identify its structures, and are used throughout this manual:

SI	SI stands for SUPERDEX index.
SI-key	Equivalent to an IMAGE/SQL search field, except in SUPERDEX the SI-key may consist of: <ul style="list-style-type: none">■ a single field (<i>simple</i> SI-key)■ a substring field (e.g. only the first 6 characters of a 12-character field or the 3rd through 6th characters).■ a combination of up to four fields or substring fields (referred to as a <i>concatenated</i> SI-key), which permits extended sorting capabilities and may be searched by the entire concatenated key value or any portion thereof.
SI-subkey	A field or substring field used as an element in a concatenated SI-key. A simple SI-key, which references only one field, has no SI-subkeys.
SI-index	The B-tree entries which are comprised of the SI-key followed by an extension which points to the corresponding data entry.
SI-extension	Included at the end of the SI-index and used to map the corresponding data entries. For entries that reside in master sets, the SI-extension consists of the full IMAGE/SQL search field value, and its length is the same as the length of the search field. For detail sets, the SI-extension is the entry's relative record number, and is two words long.
SI-path	In IMAGE/SQL, a path defines the relationship between a master and detail dataset. In SUPERDEX, an <i>SI-path</i> defines any field (or combination of fields) that can be searched via SUPERDEX, as an IMAGE/SQL path would be used to index into a detail set. Entries along an SI-path are logically maintained in alphabetical order, so an SI-path may be thought of as a virtual sorted chain containing all the entries in the dataset.
SI-chain	In IMAGE/SQL, a chain is comprised of all the entries in a detail set that have the same search field value, as specified in the DBFIND <i>argument</i> . In SUPERDEX, an <i>SI-chain</i> is a virtual chain consisting of all the qualifying entries in a master or detail set that meet the search criteria as specified in SUPERDEX's DBFIND <i>argument</i> (which may or may not have the same IMAGE/SQL search field value).
SI-subset	Used only when performing Relational Access (Boolean operations) against multiple values for a single SI-path, multiple SI-paths, datasets, and databases by performing successive DBFINDs. Both a virtual <i>active SI-subset</i> and <i>backup SI-subset</i> are maintained to contain the SI-chains retrieved by the DBFINDs.

SI-link	<p>Used when performing Relational Access against multiple datasets; it defines the common item used to logically link the different sets.</p> <p>The SI-link may also be used to enforce a sorting order when performing relational access against multiple SI-paths, sets, and bases.</p> <p>It is required that the item assigned as the SI-link be configured as SI-subkey in a concatenated SI-key; alternately, for SI-paths against a master dataset, the SI-link may be the IMAGE/SQL search master field.</p>
SI-counter	Optional parameter for the ! <i>list</i> construct for DBGET which specifies how many SI-indices should be returned with a single DBGET call.
SI-definitions	Information about the SI-paths configured for a database.
SI-dataset(s)	<p>One or more standalone detail datasets in each SUPERDEX'ed database which contain all the SUPERDEX B-tree structures. The <i>root</i> SI-dataset (named SI or SI0) contains the SI-definitions.</p> <p>For large databases or to optimize throughput, up to eight SI-datasets may be allocated, although one SI-dataset is normally sufficient. These datasets are named SI optionally followed by a sequence number (i.e. SI - SI7).</p> <p>When a database is enabled for indexing, the SI-dataset(s) will not be visible from DBINFO. Products, such as QUERY, will not display the SI-dataset(s).</p>
SI-item	The only field in the SI-dataset(s), which is configured as a compound item named SI .
SI-index base	A separate database that contains the SI-indices, which may optionally be configured for any base. With this option, all SI-indices are maintained in the separate <i>SI-index base</i> rather than the primary base.
<hr/> <p style="text-align: center;">❖ SI-index base is no longer supported with the SUPERDEX TPI Version.</p> <hr/>	
SI-pointer	A pointer in each B-tree that can be positioned before or after any index in the tree.
SI-intrinsics	The SUPERDEX intrinsics used by IMAGE/SQL which are contained in the SUPERDEX XL (XLSUPRDX.PUB.SYS).
Substring field	A partial definition of a IMAGE/SQL field. The starting character position and number of characters can be specified. This is also known as Byte Offsets and Byte Lengths.

CHAPTER 6:

Access principles

Adding, updating, and deleting entries

In SUPERDEX, entries are added, updated, and deleted using IMAGE/SQL DBPUT, DBUPDATE, and DBDELETE intrinsics. IMAGE/SQL completely supports SUPERDEX and its indices.

Qualifying and retrieving entries

Entries are qualified and retrieved using the DBFIND and DBGET intrinsics. These intrinsics now have extended capabilities and additional *modes*.

With SUPERDEX, DBFIND *mode 1* may be called against a master or detail set with an *argument* that contains multiple values, conditional and Boolean operators. The qualifying number of entries is returned in the *status* array, and an internal SI-pointer is set in the B-tree. If the qualifying entry count is not needed, it is more efficient to instead use DBFIND *mode 21*. Additional DBFIND *modes* are available to perform specialized functions, such as setting a pointer to the alphabetical first or last entry in the set.

DBGET *mode 5* may be used to retrieve the entries in ascending sorted sequential order; DBGET *mode 6* in descending order. When all qualifying entries have been returned, an end-of-chain (or beginning-of-chain) condition is returned. New DBGET *modes 15* and *16* may also be used to continue retrieving entries that are not on the SI-chain (those that no longer meet the search criteria).

Indexed access vs. relational access

Internally, one of two access methods is used in qualifying entries with DBFIND: *indexed access* or *relational access*.

Indexed access is used for retrievals that can be accomplished by accessing a single SI-chain. This accounts for most retrievals, and is used by default.

Relational access is used for Boolean retrievals that require the use of multiple SI-chains, such as in performing retrievals against multiple SI-paths, sets, and bases by using multiple DBFIND calls.

CHAPTER 7:

Compatibility

Data types

SUPERDEX handles data as stored based on the IMAGE/SQL item data types. Search values may be represented in the same format as internally stored (ASCII, binary, etc.), or special conversion operators may be used.



Unsigned and signed values for items of data type P and Z are treated identically when qualifying entries.

There are special index types (ASK Date and PowerHouse Date) that allow special handling for the normal binary items.

Status array

All SUPERDEX status information is returned in the standard IMAGE/SQL *status* array. The qualifying number of entries from DBFIND is returned in words 5-6 (like IMAGE/SQL), and end-of-chain and beginning-of-chain conditions are returned as condition words 15 and 16 in word 1 of the *status* array.

Error and exceptional condition handling

All errors and exceptional conditions are indicated by standard IMAGE/SQL error messages in the condition word field (word 1) of the *status* array.

Application programs

Existing application programs require no changes or minor, straightforward modifications to utilize SUPERDEX's capabilities.

The simplest introduction of SUPERDEX is to replace all automatic master sets with SI-paths, which makes it possible to access the records in the related detail sets generically and in sorted order without any program modifications. The user need only include an @ and/or ? in the value being searched for. Another simple modification is to replace sorted IMAGE/SQL paths with concatenated SI-keys.

Fourth-generation languages

SUPERDEX supports several 4GLs via special interfaces. VISIMAGE, TRANSACT and PROTOS either contain their own SUPERDEX interface or do not require a special interface. SUPERDEX interfaces are available for Cognos' PowerHouse products QUIZ, QTP and QUICK. Interfaces also exist for Speedware, and Sydes' SYDAID. Interfaces to other 4GLs are currently being developed.

Native Language support

SUPERDEX fully supports Hewlett-Packard's Native Language Support facility for matching, collating, and other significant operations.

7-bit support for Swedish language



A special version of SUPERDEX has been developed for support of the Swedish language.

Since new files must be supplied, please contact Bradmark if you need this version.

SECTION 2:**Access methods****Overview**

This section looks at various methods available in SUPERDEX for accessing entries in IMAGE databases.

Each chapter covers a different access method and discusses the functionality provided, its applications, details about configuration and implementation, rules of operation, efficiency, and maintenance considerations. Examples are given throughout.

Chapter 1 Multiple keys in master and detail datasets

Function For accessing data entries in manual master and detail datasets by any number of keys.

Chapter 2 Concatenated SI-keys containing multiple fields

Function Up to four fields or substring fields may be concatenated together to form a composite key.

Chapter 3 Sorted sequential retrieval

Function All entries are returned in ascending alphabetical or descending alphabetical order.

Chapter 4 Keyword retrieval

Function Permits an entry to be accessed by any significant word contained in any keyworded key.

Chapter 5 Generic and partial key retrieval

Function Allows entries to be searched for by the first few letters of a key or by a string embedded in a key.

Chapter 6 Approximate match retrieval

Function SUPERDEX can find the nearest matching entry if no entry that matches a specified value exists.

Chapter 7 Greater-than, less-than, and range retrieval

Function Permits searches for all entries that are greater than or equal to, less than or equal to, or not equal to a specified value, or that fall within the range of two values.

Chapter 8 Grouped retrieval

Function For handling multiple fields that are functionally equivalent as one logical field at lookup.

Chapter 9 Super-grouped retrieval

Function For allowing master entries to be qualified based on their contents and the contents of their related detail dataset entries.

Chapter 10 Relational access: multiple criteria retrieval

Function Permits multiple values to be specified for a field in a single lookup operation and the results to be combined by using Boolean operations.

Chapter 11 Relational access: multiple fields, sets, and bases

Function Like multiple criteria retrieval, permits multiple values and Boolean operators to be specified--but extends these capabilities to work on multiple fields, datasets, and databases.

Chapter 12 Custom indexing

Function Indices may be calculated by a user-written procedure, providing complete flexibility in indexing entries.

Chapter 13 Independent indexing

Function Permits non-IMAGE data to be indexed by SUPERDEX, for purposes such as document management.

CHAPTER 1:**Multiple keys in master and detail datasets**

Functionality

IMAGE lets you access a master set by only one field and a detail set only via its related masters unless time-consuming serial reads are used, forcing rigid applications and cumbersome database structures. SUPERDEX lets any dataset be accessed quickly by any field or fields--even every field.

Application

Due to the inability to specify more than one field in accessing a master dataset and to the high overhead of detail dataset paths, access to datasets is typically restricted to either one or a few fields.

For example, customers stored in a master dataset may be accessed only by the customer number (unless a serial read is performed). In SUPERDEX, they could be accessed by various fields, such as COMPANY-NAME, CONTACT-1, CONTACT-2, and PHONE. If the customers were instead contained in a detail dataset with several related automatic master datasets, they could stay there, and the related automatic master sets could be replaced by SI-paths with the same names as the search fields and accessed via DBFIND *mode* 1. Hence, no program modifications would be required to facilitate partial-key access and other powerful retrievals.

Index Substrings

All character type indexes (IMAGE type X or U) allow index substrings. This means indexes can begin on byte offsets from the IMAGE item and have a byte length. For example, an index can begin in position 3 of the IMAGE item for a length of 5 bytes.

Implementation

Each field or combination of fields that is to be accessible as a key must be configured in an SI-path, and SUPERDEX creates a B-tree for each one.

For simple, single-field SI-keys, the entire field could form the index or a substring of the field could be used.

Operation

DBPUT, DBUPDATE, and DBDELETE automatically maintain the SI-indices in the B-trees.

Entries are qualified by DBFIND *mode* 1 or one of several new *modes* and retrieved by DBGET *modes* 5 and 6 or new *modes* 15 and 16. If the *item* parameter contains the name of an SI-path, the B-trees are automatically accessed; otherwise, standard IMAGE access is performed.

Efficiency

It is recommended that SI-keys be kept as short as possible for efficiency. The longer the SI-key, the more access will be necessary to manage the associated SI-indices. Substring SI-keys should be used where possible. For example, 10 characters of a 20-character LAST-NAME field may be sufficient for indexing purposes, or only the 5th character through the 9th character of a general ledger account number may be sufficient.

CHAPTER 2:**Concatenated keys containing multiple fields**

Functionality

IMAGE restricts a key to a single field. SUPERDEX permits multiple fields or substring fields to be concatenated together and retrieval to be done on the entire concatenated value or any portion thereof.

Up to four fields, or substring fields, may be concatenated to form an SI-key, permitting enhanced retrieval and sorting capabilities and eliminating the need for sorted IMAGE paths and many programmatic sorts. More than four fields may be included in a concatenated key as long as they are physically contiguous in the dataset.

Application

Concatenated SI-keys permit very specific lookups to be performed without having to read down a chain to qualify entries that match on multiple fields because all fields may be contained as SI-subkeys in the SI-key.

For example, a customer number and order date could be concatenated together to form an SI-key. To access the SI-key, a single composite value would be specified and the corresponding entry returned. The alternative in IMAGE would mean reading down the customer's chain until the order with the specified date was encountered.

Index Substrings

All character type indexes (IMAGE type X or U) allow index substrings. This means indexes can begin on byte offsets from the IMAGE item and have a byte length. For example, an index can begin in position 3 of the IMAGE item for a length of 5 bytes.

Implementation

The combination of fields that are to be accessible as a concatenated SI-key must be configured in an SI-path, and SUPERDEX creates a single B-tree for the concatenated SI-index.

❖ **If related to a detail set, a concatenated SI-key may consist of up to four fields or substring fields. For master sets, up to three fields (four if the IMAGE search field is included) may be defined. Each field or substring field is called an *SI-subkey*.**

❖ **If more than four fields are needed for a concatenated key, SUPERDEX is able to support this, so long as the extra fields are contiguous in the dataset.** This is accomplished when configuring the SI-path by declaring an SI-subkey length that exceeds the length of the specified field. This results in the specified number of characters being included in the SI-subkey, thereby forming an SI-subkey that contains multiple fields or truncated fields. To utilize this feature, it is necessary to set a special JCW named SIEXTLEN during SIMAINT operation.

Operation

DBPUT, DBUPDATE, and DBDELETE automatically maintain the SI-indices for concatenated SI-keys.

Entries are qualified by DBFIND *mode* 1 or one of several new *modes* and retrieved by DBGET *modes* 5 and 6 or new *modes* 15 and 16. If the *item* parameter contains the name of an SI-path, the B-trees are automatically accessed; otherwise, regular IMAGE access is performed. The entire combined SI-key value or a partial value may be specified in the *argument* parameter.

If the concatenated SI-key has been configured using the **SIEXTLEN** JCW to include more than four SI-subkeys, it is required that all fields that are included in the concatenated SI-key but are not explicitly referenced by name be included in the DBGET *list* in the order in which they occur in the dataset, in preparation for DBUPDATE and DBDELETE.

Efficiency

It is recommended that each SI-subkey in an SI-key be kept as short as possible for efficiency. Substring SI-subkeys should be used where possible. Concatenated SI-keys are generally less efficient than simple SI-keys because they typically have longer lengths and therefore cannot be managed as efficiently; however in many instances they can outperform simple SI-keys because they can significantly reduce the number of entries qualified. Concatenated keys can also eliminate the need for sort items and programmatic sorts, again, increasing efficiency.

CHAPTER 3:**Sorted sequential retrieval**

Functionality

IMAGE returns detail chain entries in chronological order, unless sorted paths are used.

SUPERDEX returns entries in ascending or descending alphabetical order, providing a natural sorting mechanism. The sort criteria may be further extended by using concatenated SI-keys, which provide more flexibility than sorted paths and without the overhead.

Application

The ability to retrieve entries in sorted sequential order eliminates the need for many or all program sorts, and requires no special handling.

The alphabetic ordering of SI-indices also permits approximate match retrieval (described later).

Index Substrings (Byte Offsets and Lengths)

All character type indexes (IMAGE type X or U) allow index substrings. This means indexes can begin on byte offsets from the IMAGE item and have a byte length. For example, an index can begin in position 3 of the IMAGE item for a length of 5 bytes.

Operation

Entries are unconditionally returned in sorted sequential order for entries qualified in indexed access mode.

In relational access mode, an SI-link may be specified in the *item* parameter of DBFIND to enforce a sorting order.

Entries are returned in ascending sorted sequential order with DBGET *modes* 5 and 15 and descending sorted sequential order with *modes* 6 and 16. All the entries in a dataset may be read in ascending or descending sorted order by calling DBFIND *mode* 100 or 500, respectively, and DBGET *modes* 15 or 16.

SUPERDEX uses HP's Native Language Support facility in returning data contained in alphanumeric (data types X and U) items for databases in which NLS is enabled, assuring that language-specific attributes (such as esstsets and umlauts) are handled properly. The language is determined from the root file of each database, and may be established in the database schema or by DBUTIL. The collating sequences used by SUPERDEX, including language-dependent variations, are documented in HP's **Native Language Support Reference Manual**.

❖ **SUPERDEX respects the sign in sorting data contained in numeric items (data types I, J, P, R, and Z), and thereby returns negative values before positive values.**

CHAPTER 4:**Keyword retrieval**



Keyword retrieval is available only in the SUPERDEX II package.

Functionality

SUPERDEX lets you access SI-keys by any significant word they contain for SI-paths configured as keyworded. For example, an entry with the SI-key value "REDUCED INSTRUCTION SET COMPUTER" could be located by the values **REDUCED**, **INSTRUCTION**, **SET**, or **COMPUTER**.

Application

Keywording is useful for indexing fields that contain multiple values, such as company names, street addresses, last/first names, part descriptions, and comments.

Implementation



Keywording can be implemented on any *alphanumeric* (data type U or X) item, for a simple or concatenated SI-key. For a concatenated SI-key, only the first SI-subkey is keyworded.

Keyworded SI-paths are configured in the SIMAINT program by appending **/K** to the SI-path name. You must specify the *keyword length*, which refers to the maximum number of **characters** out of each keyword (not the entire IMAGE item) to include in the SI-key. For indexing purposes, words that are longer than the keyword length are truncated; those that are shorter are padded with spaces.

Also specified is the *minimum number of characters per keyword*, which defines the minimum number of characters, between 1 and 4, that a word must contain in order to qualify for keywording. This permits very short words to be easily excluded based on their length.

Additionally for each keyworded SI-path, the *average number of keywords* must be specified, which refers to the average number of significant, unique keywords within each SI-key, between 1 and 16. If requirements change at a later time, the average number of keywords may be changed by reorganizing the SI-path--a new value may be entered at that time.

To eliminate unnecessary or common words by value from keywording, an exclusion list can be defined which restricts the keyword entries to only relevant ones. Exclusion words are specified in a disk file via any editor and uploaded into a special standalone SI-path called **KWEXCLUDE**.

Operation

For SI-paths that are defined as keyworded, every word in the SI-key separated by spaces or special characters is treated as a keyword. **(You may optionally specify up to 15 special characters to be excluded as keyword delimiters when configuring SI-paths.)** Multiple SI-index entries (one for each unique value in the SI-key) are automatically generated by DBPUT and removed by DBDELETE. For compound IMAGE items that are keyworded, each subitem is examined separately and keyworded accordingly. **For concatenated SI-keys, only the first SI-subkey is keyworded.**

❖ **All keywords are upshifted for indexing and matching purposes.**

Keyworded fields are always searched by individual significant words during the DBFIND operation. Additionally:

- words that contain a hyphen are keyworded multiple times: once for each hyphen plus one. For example, "HEWLETT-PACKARD" would be multiply indexed and could be located by both **HEWLETT-PACKARD** and **PACKARD**, and "TIC-TAC-TOE" could be located by **TIC-TAC**, **TAC-TOE**, and **TOE**. (This multiple-indexing feature can optionally be disabled when configuring SI-paths.).
- SI-keys in which the same word appears more than once are indexed only once for that word.
- a maximum of 16 keywords per SI-key (for simple SI-keys) or SI-subkey (for concatenated SI-keys) is allowed.

Keyworded SI-paths are accessed in the same way as non-keyworded SI-paths--the only difference is in the configuration of the SI-path.

Efficiency

Keyword lengths should be kept as short as possible, typically 5 or 6 words, for efficiency. The minimum keyword length should be set at 4, if possible, to exclude very short words that contain less than four characters.

Commonly occurring special characters should be excluded as keyword delimiters to avoid unnecessary indexing. For example, if keywording entries in which dates are common (e.g. "02/20/90"), the slash character (/) should be excluded.

If multiple indexing of hyphenated values is not required in order to locate entries, this feature should be disabled. This is especially significant for SI-keys in which hyphens are prevalent, such as part numbers (e.g. "123-999-447").

Also, only very common words should be configured for exclusion, since the required overhead when entries are added increases with the number of excluded words.

Maintenance

All keyworded SI-paths and the **KWEXCLUDE** (exclusion word) SI-path must be reorganized whenever any changes are made to the file of excluded words.

❖ **The keyword exclusion file must be present in the same group/account as the database when SI-paths are reorganized.**

CHAPTER 5:**Generic and partial key retrieval**

Functionality

IMAGE will not find an entry unless you specify its key value exactly in its entirety. SUPERDEX permits a partial key or keyword to be specified, as well as a generic key containing wildcards.

Application

Probably the most requested capability for IMAGE databases is generic and partial key access: the ability to specify only a few significant characters of the key rather than its entire value.

This saves not only time and keystrokes, but locates entries whose exact values are not known or which cannot be located due to misspellings or other reasons.

Generic key access permits values that match a specified pattern to be located, useful for selecting entries with commonalty. Partial key access allows for a variable number of positions to be defined.

Index Substrings

All character type indexes (IMAGE type X or U) allow index substrings. This means indexes can begin on byte offsets from the IMAGE item and have a byte length. For example, an index can begin in position 3 of the IMAGE item for a length of 5 bytes.

Implementation

Generic and partial key retrievals may be performed on any *alphanumeric* field (data type X or U) referenced in an SI-path. They may not be performed against numeric fields (data types I, J, K, P, R and Z).

Operation

Partial key access can be performed by three different methods:

The first is to specify the partial key value appended with an @ as the *argument* for DBFIND *mode 1* (e.g. HEWL@). DBFIND will locate all entries that match on the significant characters followed by anything. A character other than @ may be designated as the wildcard character when configuring SI-paths.

The second method is to specify up to two (2) @ in the *argument*, surrounded by << >> for DBFIND *mode 1* (e.g. <<H@L@T>>). DBFIND will qualify all entries that contain all three groups of significant characters in the specified order.

The last method is to specify the value in the *argument* without an @ but vary the *mode* based on the length of the *argument*. For example, an *argument* containing the partial key **ROLA** would dictate *mode* 102 or -104 (100 plus the number of words or bytes, respectively, in the value). No matchcodes are allowed in this method.

Generic key retrieval is accomplished by embedding the **?** or **#** matchcodes in the *argument*.

The **?** holds the place of **ANY** character. For example, the *argument* **L?TTER** would locate "LETTER" and "LITTER"; by appending an @ (**L?TTER@**), "LETTERMAN," "LITTERBUG," and "LOTTERY" would also be located. A character other than **?** may be designated as the matchcode when configuring SI-paths, or the single-character matchcode may be disabled.

The **#** holds the place of only numeric characters. The *argument* **AP#J@** would locate "AP2J8A99" and "AP7JIT", but not "APYJ97K".

Efficiency

Search *arguments* that contain one or more **?**s or **#**s in the leftmost character positions or contain an @ in the first character position are less efficient than those that begin with alphanumeric characters. Therefore, for best performance, a substring field should be specified.

CHAPTER 6:**Approximate match retrieval**

Functionality

Neither IMAGE nor SUPERDEX can find an entry that does not exist, but SUPERDEX can do the next best thing: find the nearest matching entry.

Application

Approximate match retrieval, like partial and generic key retrieval, is useful in circumstances in which the exact key value is not known. Unlike partial and generic key retrieval, approximate match retrieval does not require that any entry matching the specified value exist: the nearest matching entry is always found.

For example, if the value **UNITED** is input and no matching entry exists, the nearest matching entry in ascending or descending order, "UNIFIED" or "UNITY," may be retrieved.

Index Substrings (Byte Offsets and Lengths)

All character type indexes (IMAGE type X or U) allow index substrings. This means indexes can begin on byte offsets from the IMAGE item and have a byte length. For example, an index can begin in position 3 of the IMAGE item for a length of 5 bytes.

Implementation

The sorted ordering of SI-indices permits approximate match retrieval by using new SUPERDEX DBFIND *modes*. If no entry that matches the search criteria exists, the internal SI-pointer is set in the B-tree to the nearest qualifying entry, permitting a program to start reading entries at any alphabetic location in either ascending or descending order.

Approximate match retrieval may be performed on any *alphanumeric* item (data type X or U) referenced in an SI-path.

Operation

Approximate match retrieval is performed by using a DBFIND *mode* that specifies how many characters in the *argument* SUPERDEX should match on, which is typically the length of the value specified. If no matching entry exists, the nearest matching entry is returned.

The *mode* also dictates whether the internal SI-pointer in the B-tree should be set before or after the matching or nearest matching entry, permitting subsequent DBGETs to include or exclude that entry.

For example, an *argument* containing the value **UNITED** would dictate *mode* 303 or -306, both of which would cause DBFIND to match on the entire value. The *mode* is calculated as 300 plus the number of words or bytes (negated if bytes). Using these *modes*, the SI-pointer would be set before the matching or nearest matching entry and select all the entries that are greater-than-or-equal-to (\geq) the value. Subsequent DBGETs in ascending order (*mode* 15) would include any entries beginning with "UNITED," while DBGETs in descending order (*mode* 16) would exclude them.

❖ **When DBFIND modes 100 to 599, inclusive, are used, DBGET modes 15 and 16 must be used.**

With a *mode* of 503 or -506 (500 plus the number of words or bytes), the SI-pointer would be set after the matching or nearest matching entry and select the entries that are less-than-or-equal-to (\leq) the value, and the "UNITED" entries would now be included with subsequent DBGETs in *mode* 16 (descending) but would be excluded with *mode* 15 DBGETs (ascending).

These modes, along with the 100 modes ($=$), 200 modes ($>$), and 400 modes ($<$) are explained in detail in under the DBFIND chapter in [Section 5: Intrinsic](#).

❖ **The modes of 100 to 599 do not support the use of matchcodes.**

CHAPTER 7:**Greater-than, less-than, and range retrieval**

Functionality

In addition to generic and partial-key retrieval, SUPERDEX permits retrievals of entries that are

- greater than or equal to a specified value
- less than or equal to a specified value
- not equal to a specified value
- in the range between two values

Application

Greater-than-or-equal-to and less-than-or-equal-to retrievals are especially useful for operations against amounts, such as finding all customers with balances of \$1000 or more.

Not-equal-to retrieval is useful for testing for the absence of a value for a particular field, such as all invoices that are not "PAID."

Range retrievals may be used against ordered values, and can be used, for example, to find all customers in a given geographical area by means of a range of zip codes. In addition, pattern matching is supported within a range retrieval which is useful, for example, for finding all orders for a given customer within a date range where the SI-key is a concatenation of the date and customer number.

Implementation

Greater-than-or-equal-to, less-than-or-equal-to, not-equal-to, and range retrievals may be performed against both alphanumeric and numeric items.



They operate on any value in any SI-key, including keyworded SI-keys.

Operation

These retrievals are performed by embedding special operators in the *argument* for DBFIND *mode* 1 or 10.

Greater-than-or-equal-to retrieval is accomplished by prefixing the argument with the **>=** operator (e.g. **>=1000**), less-than-or-equal-to retrieval uses the **<=** operator as a prefix, and not-equal-to retrieval uses the **<>** operator as a prefix. The **<>** operator can also appear after another value in the same *argument* to exclude records (e.g. **SUPER@<>SUPERDEX**).

Range retrievals are performed by using the `>=` and `<=` operators in combination. For example, a range search to find all the entries with amounts between 500 and 1000, inclusive, is specified with the *argument* `>=500<=1000`. Pattern matching may be done within a range by specifying the pattern, start point, and endpoint in the *argument*; for example, an *argument* of `??????4433>=890101@<=891231@` against a concatenated key containing date and customer number would find all the orders for the customer 4433 placed in 1989.

Entries may be retrieved in ascending or descending sorted order with DBGET *modes* 5 and 6, which return end-of-chain and beginning-of-chain conditions when all entries have been read.

Greater-than-or-equal-to and less-than-or-equal-to retrievals may alternately be accomplished without specifying the `>=` and `<=` operators and instead using any DBFIND *mode* and *argument* followed by DBGETs with new modes 15 and 16, which perform greater-than ascending and less-than descending retrievals, respectively.

CHAPTER 8:**Grouped retrieval**



Grouped retrieval is available only in the SUPERDEX II package.

Functionality

IMAGE can search only one field at a time. SUPERDEX lets you group multiple fields in a dataset together at configuration time, and automatically searches them all at lookup time, thereby handling them as one logical field. By default, SI-keys are not grouped.

This grouping technique is automatically imposed on all compound IMAGE items used in SI-keys, and generates a separate SI-index for each subitem value. The result is that every subitem is always searched automatically whenever the item is referenced.

Application

Grouping is useful for logically combining multiple fields in a dataset that are functionally identical.

For example, a two-line address may be stored in the fields ADDRESS-1 and ADDRESS-2 with addresses contained on either or both lines. The two fields may be configured as SI-keys and grouped together in an SI-path called ADDRESS, and both will be searched automatically whenever ADDRESS is referenced. Even greater functionality can be achieved by defining the grouped SI-path as keyworded, permitting any word within the address lines to be specified for retrieval.

Or, if a quick customer lookup mechanism is needed in which either the company name, contact name, or phone number may be specified in response to a single prompt, the fields COMPANY, CONTACT, and PHONE could be grouped together under the SI-path name QUICK-LOOKUP.

Implementation

Multiple SI-keys may be grouped together into a single SI-path as a configuration option in SIMAINT by appending /G to each SI-path name, except for the first.



All SI-keys contained in a group must be of the same data type. It is also required that each SI-key in the group be configured with the same length, so it is possible that some SI-keys must be truncated and others padded with spaces. For example, if COMPANY is X30, CONTACT is X20, and PHONE is X14, these three fields may be grouped together with any length between 7 and 15 words--with 7 words, the SI-key for both COMPANY and CONTACT would be truncated; with 15 words, both CONTACT and PHONE would be padded with spaces; or any length in between could be chosen.

In configuring a group, specify the longest SI-key first. It will establish the length of the grouped SI-path, as its length (or substring length, if specified) is unconditionally applied to subsequently configured SI-keys belonging to the same group.



SI-keys that are used as IMAGE master set search fields should be specified last.

Concatenated SI-keys can also be grouped. The second through the fourth SI-subkeys are repeated for each SI-key in the group. This allows COMPANY and CONTACT to be grouped and to have the LAST-ACTIVITY-DATE concatenated with both indexes.

Operation

Whenever the group is referenced by its SI-path name in the *item* parameter of DBFIND, all SI-keys that form the group are unconditionally searched.

Grouped SI-paths are accessed in the same way as non-grouped SI-paths--the only difference is in the configuration of the SI-path.

There may be some ambiguity in searching by an SI-key in a grouped SI-path whose item length is shorter than the group length and which is therefore padded with spaces. For example, if CITY, an X16, and STATE, an X2, are grouped together with an SI-key length of 8 words (to accommodate CITY), an *argument* of **CA** would find not only all entries in the state of "CALifornia" but also those in the cities of "CALABASAS" and "CARLSBAD." To resolve this ambiguity, use DBFIND *mode* 1 and pad the *argument* with enough trailing spaces to cover the full SI-key length.

CHAPTER 9:**Super-grouped retrieval**

◆ Super-grouped retrieval is available only in the SUPERDEX II package.

Functionality

IMAGE can search only a single field in a single dataset at a time. SUPERDEX lets you group together a master dataset with one or more of its related detail sets at configuration time, and automatically searches the configured fields in the related detail datasets at lookup time.

The result is that entries in master datasets can be qualified based on the values in related detail sets.

Application

Super-grouping is useful for qualifying master entries based on a logical combination of each master entry and its related detail entries, or on just the related detail entries.

For example, a library system may contain the title of a book in a master dataset, a description of the book on multiple entries in a related detail set, and the book's author(s) on one or more entries in another related detail set. The master set is keyed on book number and pathed to the two related detail sets on the same item. Together, the master and detail entries of a given book number form a profile of the book.

A super-grouped SI-path called *BOOK-PROFILE* could be configured based on the book name (from the master dataset), description (from one detail dataset), and authors (from the other detail dataset), permitting a book to be qualified by one or more of its characteristics (such as author plus title). The greatest functionality would be achieved by defining the super-grouped SI-path as keyworded, permitting any word in the book name, summary, or author--or any combination thereof--to be specified for retrieval.

It is not required that the master entry itself be contained in the super-group: it is possible to define a super-group in which the detail entries only are used to qualify their related master entries. This means that a DBFIND against the master dataset can be qualified by values only in the detail dataset.

The super-group in the previous example could have alternately been defined to consist of only the descriptions and authors (excluding the book name). This way, books could be qualified by author and/or description.

Implementation

Multiple SI-keys in related datasets may be grouped together into a single SI-path as a configuration option in SIMAINT by first defining the SI-path for the master dataset and then referencing this SI-path (by appending /G to the SI-path name) for each detail dataset that should be included in the super-group.

If instead defining a super-group that does not include the master entry itself, the SI-path is specified for the master set using an item from one of the related detail sets (since no item from the master set is included in the super-group).

❖ **All detail sets included in the super-group must be related to the master dataset by an IMAGE path. Also, the name of the item in each detail dataset that forms the IMAGE path must be the same as the name of the search field in the master dataset.**

Concatenated SI-keys can also be super-grouped. The second through the fourth SI-subkeys are repeated for each SI-key in the super-group.

Operation

DBPUT, DBUPDATE, and DBDELETE to all datasets configured in the super-group automatically maintain the SI-indices.

Super-groups may be accessed only via the master dataset--not the related detail sets. So DBFIND must be called against the master set, referencing the name of the super-group in the *item* parameter. DBFIND automatically searches all SI-keys in all (master and related detail) datasets in the super-group and qualifies the corresponding master entries based on them.

Only qualifying master entries are returned by SUPERDEX's DBGET--not their related detail entries. If desired, use standard IMAGE DBGETs (mode 5 or 6) to read the IMAGE chains to retrieve the detail entries related to the qualifying masters.

As illustrated, **SUPERDEX's DBFIND and DBGET qualify and retrieve only master entries.** The detail entries in the super-group are only used as criteria for qualifying their related master entries.

Maintenance

Whenever the SIMAINT program is used to reorganize or delete any SI-key contained in the group, all SI-keys in the group are automatically reorganized or deleted.

CHAPTER 10:**Relational access: multiple criteria retrieval**

❖ **Relational access retrieval capability using multiple criteria is available only in the SUPERDEX II package.**

Functionality

SUPERDEX can search an SI-path for a combination of multiple search criteria in a single operation.

Retrievals against multiple criteria can be used to locate entries:

- that meet either criterion (Boolean **OR** operation)
- that meet both criteria (Boolean **AND** operation)
- that meet one criterion but not the other (Boolean **AND NOT** operation)

Application

Often, it is not enough to be able to specify keys in partial or generic format; rather, it is necessary to locate entries that meet multiple criteria.

Boolean operations provide the most powerful and flexible search capability. Some examples would be finding all entries in a keyworded SI-path that contain both of two keywords (AND operation), one keyword or the other (OR operation), or one keyword and not the other (AND NOT operation).

Implementation

Boolean operations may be specified against any SI-path, regardless of its configuration.

Operation

Boolean operations are accomplished by embedding the appropriate notation in the DBFIND *argument*. Although relational access is available in DBFIND *mode* 1, DBFIND *mode* 12 can also be used and is suggested. If DBFIND *mode* 1 is used, the tilde (~) must be specified in the first position of the *argument*, and, as with DBFIND *mode* 12, the *argument* must contain a semicolon (;) in the last position.

For example, if DBFIND *mode* 12 is used in a search for all part descriptions that contain both the words PAPER and CLIP, then an argument of **PAPER AND CLIP;** could be specified. To find all invoices that are unpaid or canceled, the *argument* would be **UNPD OR CANC;**. Additionally, to find all entries in California and not Los Angeles the argument would be specified as **CA NOT "LOS ANGELES";**.

Values stored in binary may be qualified by specifying the search values in ASCII format or by utilizing the DBFIND *mode* 11.

For a more complete description of Boolean operations, refer to the *Qualifying entries with DBFIND* chapter in the *Programming* section of this manual.

Efficiency

When performing Boolean operations using multiple values, it is always recommended for efficiency to specify the less common value or values first. For example, an *argument* of **JOHN AND BROWN;** causes SUPERDEX to select all the entries that contain "JOHN" and then de-select those that do not contain "BROWN". If there are more records with JOHN, it would be considerably faster and more efficient to specify **BROWN AND JOHN;** instead, since far fewer entries would be selected in the first lookup.

CHAPTER 11:**Relational access: multiple fields, sets,
and bases**

◆ **Relational access retrieval capability using multiple fields, sets, and bases is available only in the SUPERDEX II package.**

Functionality

Relational queries may be performed using multiple values across multiple fields, datasets, and databases using dynamically-joined indices. This provides the power of a relational database within the standard IMAGE environment.

Application

This feature permits entries to be located by multiple criteria on multiple fields in a dataset, as well as using multiple datasets and databases to qualify entries.

For example, to find all unpaid invoices in the ORDERS database with amounts greater than \$1000 might require testing both the PAID-FLAG and ORDER-TOTAL-AMT fields of the INVOICE-HEADER dataset.

To find all those invoices only for customers with poor payment history would also require a lookup in the AVG-DAYS-TO-PAY field in the CUSTOMER dataset. And to attain more complete information about these customers may require access to entries archived in the ORHIST database.

Implementation

Entries are qualified based on multiple fields, sets, or bases with multiple DBFIND calls against any SI-paths. No special SI-path configuration is required.

Operation

Multiple DBFIND calls are performed in succession, with one DBFIND per SI-path with varied *base*, *dset*, and *item* parameters which specify the database, dataset, and SI-path to access.

In the preceding example, four DBFINDs would be performed in succession: one against the *PAID-FLAG* SI-path in the *ORDER-HEADER* set in the *ORDERS* base, another against the *ORDER-TOTAL-AMT* SI-path, another against the *AVG-DAYS-TO-PAY* SI-path in the *CUSTOMERS* set, and the last against the dataset and SI-path of the same name but in the *ORHIST* database.

CHAPTER 12:

Custom indexing

Functionality

SUPERDEX indexes each entry based on its configured SI-paths using the literal value of each SI-key, as influenced by truncated fields, concatenated SI-keys, and keyworded SI-paths.

There are circumstances in which this may not be sufficient to properly index an entry and where additional intelligence is required to compose the SI-key.

To address these requirements, SUPERDEX allows entries to be indexed by any value that may be calculated from the data entry.

Application

Some examples of requirements for customized SI-indices are:

- data type conversion (for ASK or PowerHouse Dates, see Section 3)
- reformatting date (e.g. ASCII to Julian)
- upshifting
- specialized SI-key extraction (embedded key)
- stripping unneeded characters
- facilitating concatenated SI-keys comprised of more than four SI-subkeys

Implementation

SUPERDEX provides an exit in the form of a user-written procedure that permits SI-indices to be calculated by parsing any values in data entries.

This procedure, named **SIUSER**, needs to be written by the user and placed in an XL that will be referenced at update time (DBPUT, DBDELETE, or DBUPDATE), but can not be located in the SUPERDEX XL (XLSUPRDX.PUB.SYS).

Operation

The SIUSER procedure is called unconditionally with every DBPUT, DBUPDATE, DBDELETE, and by the SIMAINT utility, and thereby automatically maintains the generated SI-indices if an SIUSER procedure exists.

Custom SI-keys should be handled as ASCII, since sorting is done using the binary or NLS representation and non-ASCII SI-keys would not sort properly. For binary SI-keys, the DBFIND *argument* must also be specified in binary format and *modes 1nn, 2nn, 3nn, 4nn, 5nn*, or *modes 1* or *21* with the full SI-key value, may be used.

CHAPTER 13:

Independent indexing

Functionality

Independent indexing describes the use of SUPERDEX to index entries contained in a structure other than an IMAGE database, such as flat MPE files. This permits SUPERDEX's advanced data qualification capabilities to be used on data external to an IMAGE database.

Application

SUPERDEX's independent indexing facility is intended to index files that supplement an IMAGE database, although an IMAGE database need only exist to house the SI-dataset in which the SI-indices are maintained.

For example, a document management system that manages separate word processing files could be implemented using independent indexing, with the SI-index consisting of the article title as the SI-key and the file name as the SI-extension. This permits an article to be looked up by its title, and its file name returned to the calling program.

Another use would be to reference BLOBS (Binary Large ObjectS). If an application needs to reference BLOBS on the 3000, or even on a PC, an Independent index can be used to index the BLOB. When a request is made the DBFIND would contain the argument(s) that reference the BLOB and the DBGET would return the file name, location (on the PC or LAN server), along with any other necessary information to retrieve the BLOB.

Implementation

Independent indexing is implemented by configuring a standalone B-tree for each independent SI-path using the SIMAINT program. The dataset is left blank, and the SI-path name and SI-index length, including the SI-extension, are defined.

❖ **With independent indexing, the type and value of the SI-extension is unknown to SUPERDEX and must be specified, since the file and data structures are designed by the user. Typically, a file name or record number is used as the SI-extension.**

Operation

The entities being indexed are added and deleted by some method that is unknown and of no concern to SUPERDEX.

SI-indices must be explicitly added and deleted using the new DBPUTIX and DBDELIX intrinsics, for which both the SI-key and SI-extension are specified in the *buffer* parameter. The database that contains the SI-indices is specified in the *base* parameter, and the SI-path in the *item* parameter. The *dset* parameter is left blank or set to **200**.

Entries are qualified and retrieved by DBFIND and DBGET. For both, the *base* parameter specifies the database that contains the SI-indices, the *dset* parameter is left blank or set to **200**, and the SI-path is defined in the *item* parameter. For DBGET, the **!** *list* is used to return the entire SI-index, including the SI-extension.

Independent SI-indices should be handled as ASCII, since sorting is done using the binary or NLS representation and non-ASCII SI-indices would not sort properly. For binary SI-indices, the DBFIND *argument* must also be specified in binary format and *modes* *1nn*, *2nn*, *3nn*, *4nn* and *5nn*, or *modes* 1 or 21 with the full SI-index value, may be used.

Maintenance

Correspondence of the SI-indices to the entities they reference is the sole responsibility of application programs. No method is provided for implicitly manipulating the SI-indices, nor maintaining their synchronization.

SECTION 3:

Configuration / Establishing SI-indices

Overview

This section describes the methods used in configuring SUPERDEX for your databases. This includes establishing the SI-item and SI-dataset(s), defining SI-paths, and establishing B-trees and SI-indices.

This section assumes that you have already loaded SUPERDEX on your system, as described in the separate *SUPERDEX loading instructions*.

Chapter 1 Installation

Function describes how to install SUPERDEX on your system, along with how to enable and disable a database for SUPERDEX.

Chapter 2 Configuration overview

Function provides a brief description of the various configuration options, including simple vs. concatenated SI-keys, handling of compound IMAGE items, SI-path names, SI-key lengths, and various restrictions.

Chapter 3 Excluding words from keywording

Function reviews the process of excluding words from keywording and shows how to create the KWEXCLUD file and configure exclusion words.

Chapter 4 Customizing default characters

Function illustrates customizing default characters to optionally redefine the characters used to represent the wildcard and matchcode, to disable multiple indexing of hyphenated values for keyworded SI-paths, and to exclude certain special characters from being recognized as keyword delimiters.

Chapter 5 Configuring SUPERDEX using SIMAINT

Function describes the methods of configuring SUPERDEX using SIMAINT and discusses program operation, access requirements, and input rules. It then gives examples of how to define various simple and concatenated SI-paths for keywording, grouping, and other functions, as well as custom and independent SI-paths.



Options for maintaining SI-paths are not discussed in this section--refer to the *Maintenance and utilities* section for information.

CHAPTER 1:

Installation

Because SUPERDEX is now imbedded in TurboIMAGE/iX, once the SUPERDEX installation tape has been loaded there are **NO** additional steps necessary for completing the SUPERDEX installation! Simply enable a database for indexing using SUPERDEX and TurboIMAGE/iX will call the SUPERDEX intrinsics.

Enabling a Database for SUPERDEX Indexing (TPI)

If an existing database is a SUPERDEX Pre-TPI database, there is **NO** conversion necessary for the indexes. The base must only be marked and enabled as a SUPERDEX TPI database. To do this simply run SIMAINT.SUPERDEX.SYS, enter the database name, and enter **RETURN** for the dataset prompt. This will mark the database as enabled for SUPERDEX TPI.

```
:RUN SIMAINT.SUPERDEX.SYS
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database >OEDB
Dataset > RETURN
```

If the database has been disabled, the same steps must be taken: run SIMAINT.SUPERDEX.SYS, enter the database name, and enter **RETURN** for the dataset prompt.

Disabling a Database for SUPERDEX Indexing (TPI)

To disable a database that is enabled for SUPERDEX, you must use DBUTIL.PUB.SYS. Run DBUTIL.PUB.SYS and enter **DISABLE basename FOR INDEXING**. This disables the database indexing. Follow the previous steps to enable a database.

```
:RUN DBUTIL.PUB.SYS
HP30391C.04.07 TurboIMAGE/XL: DBUTIL (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1987

>>DISABLE OEDB FOR INDEXING

  Indexing is disabled for SUPERDEX.

  Indexing is disabled.
>>exit
```

CHAPTER 2:**Configuration overview**

Defining SI-keys and SI-paths

SI-paths are defined in the **SIMAIN**T program, which initially creates the B-tree structures and optionally the SI-item and SI-dataset(s).

The SIMAIN

T program may need to know any of the following, which are discussed in detail on the following pages:

- the dataset to which each SI-path is related
- the SI-dataset that contains the SI-indices for a dataset's SI-paths
- the name of each SI-path
- the field or fields that form each SI-key and their lengths
- whether each SI-path is keyworded, grouped, supergrouped, etc...
- whether to index entries that contain blank values in the first SI-subkey, for each SI-path
- the keyword length, for each keyworded SI-path
- the minimum number of characters per keyword, for each keyworded SI-path
- the average number of significant words that will be contained in the SI-key, for each keyworded SI-path
- the keyword exclusion length, for all keyworded SI-paths
- the SI-key length, for each custom SI-path
- the SI-index length, for each independent SI-path
- up to 15 special characters to optionally exclude as keyword delimiters
- an optional replacement character for the @ wildcard/terminator
- an optional replacement character for the ? matchcode operator
- an optional replacement character for the # matchcode operator
- whether to disable multiple indexing of hyphenated values for keyworded SI-paths

Naming SI-paths

The name of the SI-path is important because it is later used in the *item* parameter of SUPERDEX's DBFIND intrinsic. Also, it may not contain a forward slash (/) unless the SI-path name is identical to an IMAGE item name that contains a forward slash.

The SI-path name must be unique within a dataset but multiple datasets may contain SI-paths of the same name, unless the SI-path is for a super-group, in which case the SI-path name must be unique within the database. It is recommended that the SI-path name not be the same as the item name that forms the SI-key or any other item, since this can cause programs that are attempting IMAGE access via the IMAGE path to instead perform SUPERDEX access via the SI-path. This may, however, be desirable, as explained below.

If an SI-path related to a dataset is given the same name as an existing item used as an IMAGE search field in the same dataset, the SI-path is used instead of the IMAGE path. If the entry cannot be found using the SI-path, the IMAGE path is used instead. If both fail, condition word 17 ("NO ENTRY") is returned. This is useful for replacing automatic master sets with SI-paths: just name the SI-path the same name as the search field in the detail dataset and programs do not require modification.

If, however, you would like to access entries alternately by an SI-path and by an IMAGE path, assign a name other than the item name to the SI-path.

Simple vs. concatenated SI-keys and their lengths

An SI-key can be *simple* (a single field) or *concatenated* (a combination of fields). The latter is useful for both searching and sorting by extended criteria.

A concatenated SI-key for a detail set may consist of up to four fields. For a master set, the search field and up to two additional fields may be defined. If more than four fields are required in a concatenated SI-key and the desired fields are contiguous, they may be included and thereby exceed the four field limit. If the search field is included as an SI-subkey in a concatenated SI-key for a master set, it must be defined last.

For a simple SI-key or for each SI-subkey in a concatenated SI-key, each alphanumeric item that exceeds one word may be included in full, or its length may be shortened (and can start in any position). This is referred to as a substring SI-key or substring SI-subkey. The exception is master dataset search fields used in SI-keys, which may not be substring.

❖ When performing relational access against multiple datasets, it is required that a common item (called an *SI-link*) from each dataset be included in an SI-key for each dataset, except for master datasets in which the common item is the search field. The recommended method for implementing this is to include the common item as an SI-subkey in a concatenated SI-key.

Compound items

Compound IMAGE items may be used in simple SI-keys or as the first SI-subkey in concatenated SI-keys, including compound items with subitems of odd-byte lengths. SUPERDEX automatically treats compound items as grouped SI-keys and generates a separate SI-index for each subitem value, the result being that every subitem is always searched automatically.

For concatenated SI-keys that include compound IMAGE items, an SI-index is automatically generated for each subitem value, with the values of the other SI-subkeys in the SI-key repeated in each SI-index. For example, for an SI-key that consists of the compound item ORDER-COMMENTS (a 5X72) and the regular item ORDER-NUMBER (an X12) five SI-indices would be created with each one containing a different ORDER-COMMENT value but the same ORDER-NUMBER.

Keyworded SI-paths

Any *alphanumeric* (data type U or X) SI-key, either simple or concatenated, may be defined as keyworded. For concatenated SI-keys, only the first SI-subkey is keyworded.

Keyworded SI-paths are configured in SIMAINT by appending **/K** to the SI-path name. Three attributes must be specified for each keyworded SI-path:

- maximum length of each keyword
- minimum number of characters per keyword
- average number of keywords in the field for each entry

The *keyword length* determines the maximum number of characters to include in the SI-key, which is independent of the length of the field. For indexing purposes, words that are longer than the keyword length are truncated; those that are shorter are padded with spaces. For example, with a keyword length of 5 words, the first 10 characters of each word would be included, so for the word "MANUFACTURING," only "MANUFACTUR" would be included in the SI-key.

❖ **It is desirable for efficiency to keep the keyword length as short as possible and not to exceed a keyword length of 6 words (12 characters).** Substring keywords can still be retrieved using the full keyword as the search *argument*, but other entries may also be returned; for example, **MANUFACTURING** would qualify entries with the value "MANUFACTURER" because only 10 characters are stored in the SI-index and the argument length is truncated to 10 characters.

The *minimum number of characters per keyword* determines the minimum number of characters that a word must contain in order to qualify for keywording. For example, the word "ASK" would be included in keywording with a minimum keyword length of 1, 2, or 3 but excluded with a length of 4. The minimum keyword length may be between 1 and 4, with 1 effectively meaning that all words are included in keywording. This value should be set to 4 wherever possible for efficiency.

The *average number of keywords* refers to the average number of significant keywords that would be contained in this field, between 1 and 16. For example, "ACME MANUFACTURING PARTS" contains three keywords. The value of this parameter is used to reserve sufficient internal space, so fractional averages must be rounded up, and it is better to estimate high instead of low if in doubt; for example, if an average of 2.5 words are contained in the field, specify a value of 3. If requirements change at a later time, the average number of indices may be changed by reorganizing the SI-path--a new value may be specified at that time.

To eliminate common words from keywording, an exclusion list may be defined, which restricts the keyword entries to only relevant words. Exclusion words are specified in a disk file using any editor and uploaded into a special standalone SI-path called **KWEXCLUDE**. You must define a keyword length for this SI-path, which is applied against all the keyworded SI-paths in the database. The keyword length may be redefined at a later time if required.

Grouped SI-paths

Multiple SI-keys related to a given dataset may be grouped together as a single SI-path, with the following restrictions:

- each SI-key in the group must be of the same data type
- each SI-key in the group will internally have the same length, which is assigned for the first configured SI-key in the group and inherited by subsequently-configured SI-keys

For concatenated SI-keys in a group:

- the second - *n*th SI-subkeys are automatically and unconditionally imposed on all SI-paths in the group
- the first SI-subkey of every SI-key in the group must be of the same data type
- the first SI-subkey of each SI-key must be assigned the same length

Because each SI-key in a group must be of the same length, it is possible that the lengths of some SI-keys will be truncated and others padded with spaces. For example, if COMPANY is X30, CONTACT is X20, and PHONE is X14, these three fields may be grouped together with any length between 7 and 15 words--with 7 words, the SI-key for both COMPANY and CONTACT will be truncated; with 15 words, both CONTACT and PHONE would be padded with spaces; or any length in between could be chosen.

SI-paths are grouped together as a configuration option in SIMAINT by appending /G to each SI-path name except the first. The length and second - *n*th SI-subkeys of the first SI-path defined are unconditionally applied to subsequent SI-paths belonging to the same group, so define the SI-path that contains the desired SI-subkeys and longest SI-key first. If one of the SI-keys in a group for a master dataset is the IMAGE search field, define it last.

Super-grouped SI-paths

A master set and one or more related detail sets--related by IMAGE paths--may be super-grouped together as a single SI-path, with the following restriction:

- the item name of the search field used to form the IMAGE path must be the same in the master set and all the necessary detail sets

For concatenated SI-keys in a super-group:

- the second - *n*th SI-subkeys are automatically and unconditionally imposed on all SI-paths in the super-group
- the first SI-subkey of every SI-key in the super-group must be of the same data type
- the first SI-subkey of each SI-key must be assigned the same length

It is not required that a field in the master set be configured as an SI-key: it is possible to configure only SI-keys in the detail sets to be used to qualify entries in the related master set. In this case, use the field in the detail set when configuring the master path. This will be used to define the length and type of path.

SI-paths are super-grouped together as a configuration option in SIMAINT by appending **/G** to each SI-path name except the SI-path related to the master set. The length and second - *n*th SI-subkeys of the first SI-path defined are unconditionally applied to subsequent SI-paths belonging to the same super-group.

Custom SI-paths

Custom indexing (maintained by the SIUSER procedure) is implemented by defining an SI-path of an arbitrary name for each custom index. Although the dataset is defined, along with the SI-path, no items are specified, since the SI-indices may not directly reference any items. The SI-key length, excluding the SI-extension, is defined, as well as the average number of SI-indices per entry.

Independent SI-paths

Independent indexing is implemented by configuring a standalone B-tree for each independent SI-path. The dataset is left blank, and the SI-path name and SI-index length, including the extension, are defined.

Blank SI-keys

By default, SUPERDEX will not generate any SI-indices for any entry that contains a blank SI-key value. For a concatenated SI-key, it will not generate any SI-indices for an entry whose first SI-subkey is blank.

This is done for efficiency and disk savings, but differs from IMAGE's method of creating a "null" chain of all blank keys. To override SUPERDEX's default and cause SI-indices to be generated for entries with blank SI-keys, append **/B** to the SI-path name.

Summary of restrictions

Most configuration options may be used in combination against all data items, although some restrictions exist. This summarizes the aforementioned restrictions:

- only alphanumeric items (data types U and X) may be keyworded
- keywording functions only for the first SI-subkey in a concatenated SI-key
- substring fields, or byte offsets, may only be defined for alphanumeric items (data types U and X) whose lengths exceed one word.
- all SI-keys in a group must be of the same data type

- all SI-keys in a group must be assigned the same length (alphanumeric fields may be truncated or padded with spaces)
- if a grouped SI-path for a master dataset contains the IMAGE master search field, it must be configured last
- concatenated SI-keys may contain a compound item as the first SI-subkey only

CHAPTER 3:

Excluding words from keywording

Definition and Purpose

To minimize disk utilization, SUPERDEX permits common words to be excluded from keywording. These exclusion words are user-specified and apply to all keyworded SI-paths in a database.

All the words that SUPERDEX should exclude from keywording for a database are entered into a file named **KWEXCLUD** and then transferred into the special Independent SI-path **KWEXCLUDE** by the SIMAINT program.

Just as each keyworded SI-path has an associated keyword length that determines how many characters of each keyword are recognized for indexing, the KWEXCLUDE SI-path also has an associated *keyword length* which is applied against each word in the exclusion file. Because the exclusion words are compared against all keyworded SI-paths, and because each keyworded SI-path may have a different keyword length, it is important to assure that the excluded words will work effectively for each SI-path. This means that only as many characters as are defined by the keyword length are matched on; therefore, it is recommended that the shortest keyword length configured for any SI-path in the database be used as the keyword length for the KWEXCLUDE path.

Exclusion words are entered into an unnumbered file named KWEXCLUD which must be located in the group/account in which the corresponding database resides. Multiple KWEXCLUD files may be used, one per database. If multiple databases reside in the same group/account and require different KWEXCLUD files, create additional files under different names and reference them with :FILE equations, for example:

```
:FILE KWEXCLUD=KWEXFILE
```

Example

Exclusion words may be entered in any editor that creates an unnumbered ASCII file, one per line, as shown in the following example. In this example, EDIT/3000 is used, but any editor may be used.

```

:EDITOR
HP32201A.07.17 EDIT/3000
(C) Hewlett-Packard CO. 1985
/A
  1      INC
  2      INCORPORATED
  3      CORP
  4      CORPORATION
  5      DIVISION
  6      ASSOC
  7      ASSOCIATES
  8      //
/K KWEXCLUD,UNN
/EXIT

END OF SUBSYSTEM

```

For this example, the keyword length that will be imposed on the KWEXCLUDE SI-path (when it is defined in SIMAINT) is 4 words (8 characters). The words on line 2, 4 and 7 exceed this length, so they will be truncated. The shorter words on lines 1, 3, and 6 will be padded with spaces, which are ignored for comparison.

Once the KWEXCLUD file has been created, it is necessary to define the KWEXCLUDE SI-path using the SIMAINT program, as described in the following chapter. Although this file may be created and modified at any time, it is recommended that it be created in its entirety before configuring any keyworded SI-paths because any changes to the file require that the KWEXCLUDE path and all keyworded SI-paths be reorganized.

Default File

The KWEXCLUD.SUPERDEX.SYS file is an example file containing common words to be excluded. Except in very special cases, it is recommended that this file be used in databases with keyworded SI-paths to reduce the amount of disk space required to create and maintain the indices. For example:

```

:FILE KWEXCLUD=KWEXFILE
:RUN SIMAINT.SUPERDEX.SYS
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database >OEDB
Dataset > space + return
SI-Path > KWEXCLUDE

```

CHAPTER 4:

Customizing default characters

SUPERDEX recognizes a few special characters that influence the operation of various intrinsics. Because of unique characteristics that your data may have, SUPERDEX allows you to redefine these characters to suit your individual requirements.

For DBFIND, SUPERDEX reserves the following special characters as conditional operators:

- @ as wildcard and terminator
- ? as a positioned matchcode
- # as numeric matchcode

For DBPUT, DBUPDATE, DBDELETE, and SIMAINT indexing on keyworded SI-paths, the following special characters are reserved:

- all special characters as keyword delimiters
- - as a keyword delimiter which results in multiple indexing of hyphenated values

So, if your system contains data values that include ?s or #s as literal characters, it would be restrictive to use the default ? or # character as a single-character matchcode; therefore, the matchcode operator could be redefined to some other character not commonly found in your data values (such as %).



When assigning replacement characters for @, #, and ?, it is important to choose characters other than those already treated specially by DBFIND as relational or Boolean operators (documented under DBFIND in the *Intrinsics* section in this manual).

Another circumstance in which default characters should be redefined is in recognizing keyword delimiters for keyworded SI-paths. By default, spaces and all special characters are treated as keyword delimiters, and it may be desirable to restrict which special characters are recognized. For example, values that include fractions ("1/2") or dates ("01/30/90") suggest that / should be excluded as a keyword delimiter; otherwise "1/2" would be indexed as "1" and "2."

Additionally, because hyphenated values are by default indexed multiple times ("01-30-90" is indexed as "01-30," "30-90" and "90") it may be desirable to disable this feature and instead treat hyphens (-) as regular keyword delimiters (resulting in "01," "30," and "90"). Or, it may be best to disable hyphens as keyword delimiters altogether.

To redefine default characters, include a customization string of up to nineteen (19) characters which defines the desired defaults as an INFO parameter when running the SIMAINT program (described later). The wildcard/terminator, alphanumeric matchcode, numeric matchcode, multi-indexing for hyphenated words, and up to fifteen (15) excluded keyword delimiters can all be included in the customization string.

The customization string specified when running SIMAINT affects all SI-paths in the database.

❖ **Once specified, the customization string is written into the internal SI-definitions for the database and need not be specified again.** If the default characters need to be changed at a later time, run SIMAINT with a new customization string; if keyword delimiters are changed, also reorganize all keyworded SI-paths.

❖ **To preserve the default values for wildcard and matchcode operators (@, #, and ?) it is necessary to specify them explicitly in the customization string.**

The nineteen (19) characters (bytes) in the customization string are represented as follows:

<i>byte</i>	<i>description</i>
1	wildcard and terminator character
2	positioned matchcode, blank disables this feature
3	numeric matchcode; blank disables this feature
4	- if multi-indexing on hyphen, blank to treat hyphen as regular delimiter
5	special character to be treated as literal rather than keyword delimiter
6	same
7	same
.	same
.	same
.	same
19	same

For example, running SIMAINT with the customization string shown:

```
:RUN SIMAINT.SUPERDEX.SYS;INFO="%^* /:-."
```

would result in the following: % is recognized (in place of @) as the wildcard and terminator character, ^ (instead of ?) is the single-character matchcode, * replaces # as the single-numeric matchcode, no multi-indexing is performed for hyphenated values, and /, :, -, and . are not treated as keyword delimiters but rather as regular literal characters. Values for positions 5 through 19 (special keyword non-delimiters) are optional and spacing the ;INFO string is not necessary.

❖ **It is necessary to specify language specific characters, such as German umlauts (Ä Ö Ü ä ö ü ß) as special characters if the database is not enabled for NLS. Otherwise they will be handled as delimiters, and words utilizing umlauts would not be retrievable (i.e. Tür).**

To display the configured customization string, run SIMAINT with the LIST entry point as shown:

```
:RUN SIMAINT.SUPERDEX.SYS,LIST
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database >OEEDB
Customization string: %^* /:-.
The following SI-Paths and items are defined:
. . .
```

CHAPTER 5:**Configuring SUPERDEX
using SIMAINT**

The SIMAINT program automatically establishes the B-tree structures and SI-indices for all SI-paths when they are created. It adds the SI-item and SI-dataset(s) for TurboIMAGE/iX databases, although when the database is enabled for SUPERDEX indexing the SI-dataset(s) will not be seen (i.e. will not be listed as an existing set by QUERY and other products using DBFIND mode 203).

SIMAINT is also used for deleting, reorganizing, and performing other maintenance functions on existing SI-paths. Options for existing SI-paths are not discussed in this section but in the *Maintenance and utilities* section, which also includes a table describing the various operations that may require SI-path maintenance.

Creating SI-item and SI-dataset(s)

The SI-item and SI-dataset(s) that are used for storing SUPERDEX's B-trees may be created by one of two methods:

- by SIMAINT.SUPERDEX.SYS, which creates them automatically
- by conventional means, such as DBGENERAL, or DBUNLOAD/DBLOAD

The preferred method is to use SIMAINT, since it is easier and automatic. After creating the SI-item and SI-dataset(s) based on the SI-path configuration, SIMAINT proceeds to create the required SI-definitions and SI-indices; however, the dataset block sizes must be even multiples of 128 words (e.g. 128, 256, 384, 512).

Operation

The SIMAINT program is prompt-driven, and includes an on-line help facility that displays datasets and the items they contain. SIMAINT operates in three discrete phases, which are automatically invoked in succession:

- *dialog* phase: all configuration information is specified
- *extension* phase: the SI-item and SI-dataset(s) are added, as necessary
- *indexing* phase: all SI-indices are generated

This permits all configuration information to be specified up front in the dialog phase and for the program to be left unattended during extension and indexing.

Access requirements

To set up a database for SUPERDEX the first time, before running SIMAINT, make sure:

- you have exclusive access to the database
- you are logged on as the database creator
- you are logged into the group and account in which the database resides

It is also recommended for performance reasons that you:

- disable ILR
- disable logging

❖ **If the database already contains the SI-item and the necessary SI-dataset(s), shared access is allowed during the SIMAINT processing (see Invoking SIMAINT,SHARED in this section).**

Input rules

These rules govern SIMAINT input:

- all input may be in upper- or lower-case
- ? displays structural help (sets and items)
- \ flushes all activity for a given level and returns to the previous level
- space returns to the previous level in the hierarchy while retaining the activity in a level
- all lengths--with the exception of *minimum keyword length*, *average number of characters per keyword* and *offset in bytes*, which are always specified in bytes--are reported and specified in words if a positive value, and bytes if a negative value. It is necessary to convert for alphanumeric (data types U and X) items (e.g. X20 = 20 bytes or 10 words).

Dialog phase

SIMAINT dialog is structured in a hierarchical fashion whereby you are led down through various levels until all required information has been supplied, and then automatically returned to the previous level. The levels in the hierarchy are:

```
Database >
  Dataset >
    SI-Path >
      Item n >
        SI-subkey prompts
```

This organization encourages a logical ordering in configuring SUPERDEX for multiple datasets and items, assures that all required information is specified, and permits an easy and consistent method for canceling and reentering input for any level in the hierarchy.

SIEXTLEN JCW for special concatenated SI-keys

A concatenated SI-key can normally contain no more than four fields as SI-subkeys. It is possible, however, to create a concatenated SI-key with more than four SI-subkeys, provided that the additional fields are contiguous in the dataset and are character fields (type X or U). If they are, it is possible to define a length for any SI-subkey that exceeds the field length, and the additional fields or truncated fields that follow are included in the SI-key, based on the length specified.

When using SIEXTLEN the additional fields should be of the same data types, the records could be returned in an unexpected sort order.

If you need to utilize this feature, it is necessary to set a special JCW named **SIEXTLEN** to **1** before invoking SIMAINT.

```
:SETJCW SIEXTLEN=1
```

Invoking SIMAINT

To invoke SIMAINT:

```
:RUN SIMAINT.SUPERDEX.SYS
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993
```

Invoking SIMAINT,SHARED

It is now possible to run SIMAINT in a shared mode. This allows tremendous availability to databases. Database Administrators can add, delete, and reorganize indexes while the database is in use. This includes on-going maintenance to the database. Exclusive access is not required on the dataset being indexed. As long as the set is not locked by another process, SIMAINT can proceed with the indexing process.

To invoke SIMAINT,SHARED:

```
:RUN SIMAINT.SUPERDEX.SYS,SHARED
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

SIMAINT executed in Shared Access Mode.
Do not execute more than one SIMAINT on the same data base at the same time!!!
```

❖ Although there can be many users on a database while SIMAINT,SHARED is being executed, only one SIMAINT,SHARED should be executed at a time on a database. This is because the SUPERDEX root information will be modified.

❖ SIMAINT,SHARED is not valid if SUPERDEX has not already been installed on the database.

Defining database

SIMAINT can be run against databases that have never been configured for SUPERDEX as well as those that already contain SI-paths.

The OEDB database in the examples through out this section contains no SI-paths.

```
Database >OEDB
Database has not been initialized for SUPERDEX
After the dialog phase the program will go into privileged mode to add the SI-
Dataset(s) to the database
```

❖ **Shared access is not available if the database has not already been initialized for SUPERDEX.**

Defining datasets

SIMAINT can be run against datasets that already have related SI-paths as well as those that do not. If a dataset already contains SI-paths, they are displayed.

Enter the name of a manual master (automatic masters are allowed for Super-Grouped SI-Paths) or detail dataset in the current database, optionally followed by one of the following suffixes:

- /n** SI-dataset to contain the SI-indices for all dataset's SI-paths, where *n* is the number of the SI-dataset between 1 and 7
- /D** Delete all dataset's related SI-paths (refer to *Maintenance and utilities* section)
- /R** Reorganize all dataset's related SI-paths (refer to *Maintenance and utilities* section)

In the following example, no suffix was specified, therefore all indices for SI-paths related to the PRFD dataset will reside in the root SI-dataset:

```
Dataset >PRFD
```

Once all datasets have been defined, hit **RETURN** at the **DATASET** prompt. This indicates to SIMAINT that you are done defining SI-paths, and causes it to proceed to the extension stage.

❖ **Once an SI-dataset has been selected for a dataset, all indices for that dataset must remain in the same SI-dataset. Different SI-paths for the same dataset cannot be placed in different SI-datasets.**

Defining associated SI-datasets

In this example, all the SI-indices for all SI-paths related to the CUST dataset will be built in the SI-dataset named **SI1**:

```
Dataset >CUST/1
```

Up to eight SI-datasets (including the root SI-dataset) may be configured for any database. Multiple SI-datasets are useful for optimizing concurrent access and necessary for large databases in which an SI-dataset may outgrow the maximum file size allowed by the MPE/iX operating system (4 Gigabytes). Valid SI-dataset suffixes are /1 through /7. In assigning suffixes, do not skip any numbers.

The suffix is automatically retained for all subsequent configuration against the dataset, and may only be changed by deleting and re-adding all the SI-paths related to the dataset. Likewise, if the dataset was originally configured without a suffix, no suffix may be specified without redefining all the dataset's SI-paths.



With super-grouping, the master and the related detail set(s) must be configured with the same SI-dataset.

Bypassing datasets for independent SI-paths

When defining independent SI-paths which are not associated with any dataset, and when defining the keyword exclusion path **KWEXCLUDE**, a **SPACE** followed by a **RETURN** is specified at the dataset prompt:

```
Dataset >space + return
```

To specify which SI-dataset will contain all independent SI-paths for the database, specify the SI-dataset number after the **SPACE**, as shown:

```
Dataset >space/2
```


Defining SI-paths and SI-keys

SI-paths may be:

- simple or concatenated
- keyworded
- an ASK Date format
- an PowerHouse Date format
- a grouping of multiple SI-keys
- a supergrouping of multiple SI-keys
- customized, for SI-keys generated via the SIUSER procedure
- standalone, for independent indexing

As already discussed, each SI-path name is arbitrary, and may be the same as an existing item name. The SI-path name may optionally be appended by one of the following suffixes:

- /K** SI-path is **K**eyworded
- /A** SI-path is an **A**SK Date index
- /P** SI-path is an **P**owerHouse Date index
- /G** SI-path consists of multiple SI-keys that form a **G**roup
- /B** **B**lank entries are indexed (by default, no SI-index is generated for an SI-key value that is blank or for a concatenated SI-key, whose first SI-subkey is blank)
- /D** **D**elete existing SI-path (refer to the *Maintenance and utilities* section)
- /R** **R**eorganize existing SI-path (refer to the *Maintenance and utilities* section)

After the SI-path name has been entered, SIMAINT prompts for the item(s) that comprise the corresponding SI-key. A simple SI-key consists of only one item; a concatenated SI-key may contain between two and four items.

For alphanumeric items (data types U or X), a truncated field rather than the full field may be included in both simple and concatenated SI-keys. For numeric items, no truncation is allowed, so the request to shorten the length is not asked.

Defining simple SI-keys

In this example, the arbitrary name *CITY* is assigned for the SI-path that consists of the full field CUCITY:

```
SI-Path >CITY/B
Item 1 >CUCITY
Item length is 8 words. Enter shorter length(- =Bytes) if desired>return
Item 2 >return
SI-Path >
```

-
- ◆ **RETURN** was pressed to accept the full field length for the SI-key and also pressed when SIMAINT prompted for the second SI-subkey, thereby defining this SI-key as simple rather than concatenated. Also, **/B** was appended to the SI-PATH, which causes SUPERDEX to override its default and generate SI-indices for entries that contain blanks in the first SI-subkey.
-

If the name specified for an SI-path is the name of a field in the dataset--such as when replacing an automatic master dataset with an SI-path--SIMAINT assumes that the first SI-subkey is that item and therefore does not prompt for it:

```
SI-Path >PRFIDF
Item length is 2 words. Enter shorter length (- =Bytes) if desired>return
Item 2 >return
SI-Path >
```

To enter a byte length, use a negative value for the length. If the length entered is not equal to the IMAGE Item length, or the length was entered as a byte length (negative value), then a byte offset can be entered. When a byte length (negative length value) is used, the index will include the requested number of characters followed by a space. The padded space is used to ensure that the sub-key is an even number of bytes in length (on a 16-bit word boundary). Note that the first character position in the item is treated as offset 1 not 0.

Defining concatenated SI-keys

To define a concatenated SI-key, specify additional fields to be included as SI-subkeys:

```
SI-Path >ORDER-ID/B
Item 1 >OMCPON
Item length is 7 words. Enter shorter length (- =Bytes) if desired >2
Enter offset in bytes (RETURN=1) >return
Item 2 >OMBLOC
Item 3 >OMCCOD
SI-Path >
```

❖ There are several items to notice in this example. First, only the first 2 words (4 characters) of the OMCPON field are being included in the SI-key. Second, since the item has been shortened, the desired offset is requested. Next, the length was not prompted for in the second and third SI-subkeys because the items are not of an alphanumeric data type and numeric items must be represented in full. Fourth, only three SI-subkeys (rather than four) were prompted for because the SI-path is being configured for a master dataset and the IMAGE search field is not included in the SI-key. Lastly, /B was specified to cause SUPERDEX to generate SI-indices for entries that contain a blank value in the first SI-subkey.

The following example defines the same concatenated SI-path, but defines the length using odd-bytes, instead of words. Additionally, an offset of 3 has been chosen. In the concatenated index, the first sub-key from OMCPON will contain three characters, followed by a space. The padded space is used to ensure that the sub-key is an even number of bytes in length (on a 16-bit word boundary).

```
SI-Path >ORDER-ID/B
Item 1 >OMCPON
Item length is 7 words. Enter shorter length (- =Bytes) if desired>-3
Enter offset in bytes (RETURN=1) >3
Item 2 >OMBLOC
Item 3 >OMCCOD
SI-Path >
```

In the following example, a concatenated SI-key containing three SI-subkeys is being configured for a detail dataset. **No SI-subkey lengths are prompted for because all items are numeric:**

```
SI-Path >ITEM-ID
Item 1 >OMNUMB
Item 2 >ITMNUM
Item 3 >ITMCOB
Item 4 >return
SI-Path >
```

❖ As explained earlier, a concatenated SI-key can contain more than four SI-subkeys only if the desired excess fields are contiguous in the dataset, are character fields (type X or U), and if the special **SIEXTLEN** JCW is set to 1.

This example shows a concatenated SI-key containing eight SI-subkeys, although only four are referenced by name. The situation is that there are eight separate fields that hold status codes--named *STATUS-1* through *STATUS-8*; and it is necessary to have all eight four-character codes included in the concatenated SI-key:

```
SI-Path >STATUS
Item 1 >STATUS-1
Item length is 2 words. Enter new length (- =Bytes) if desired >return
Item 2 >STATUS-2
Item length is 2 words. Enter new length >return
Item 3 >STATUS-3
Item length is 2 words. Enter new length >return
Item 4 >STATUS-4
Item length is 2 words. Enter new length >10
Enter offset in bytes (RETURN=1) >return
SI-Path >
```

- ❖ The default length of two words was chosen for the first three status codes (by pressing **RETURN**). A length of 10 was specified for *STATUS-4*, which will cause the remaining status codes (*STATUS-5* through *STATUS-8*) to be implicitly included in the SI-key. Also, the prompts displayed by SIMAINT are not the same when the **SIEXTLEN JCW** is turned on.

Remember that in order to configure this special type of concatenated SI-key, it is necessary to have the **SIEXTLEN JCW** set to 1 before invoking SIMAINT.

Defining keyworded SI-paths

Appending **/K** to the SI-path name defines an SI-path that is keyworded:

```
SI-Path >CUNAME/K
Item length is 15 words. Enter keyword length>6
Enter minimum number of characters per keyword (1-4) >4
Enter average number of keywords per entry >3
Item 2 >return
SI-Path >
```

- ❖ Instead of prompting for the SI-key length, SIMAINT instead prompts for keyword length, minimum number of characters per keyword, and average number of keywords. The keyword length can be entered in bytes as a negative value. Also, the offset cannot be specified, since the entire field is included in the keyword.

In this example, each keyword is recognized only by the first 12 characters (6 words), a word must contain at least 4 characters to be keyworded, and an average of 3 keywords are contained in each SI-key.

Concatenated SI-keys can also be configured as keyworded, but only the first SI-subkey is keyworded:

```
SI-Path >SHIP-ID/K
Item 1 >SHNAME
Item length is 15 words. Enter keyword length >-12
Enter minimum number of characters per keyword (1-4) >3
Enter average number of keywords per entry >4
Item 2 >SHADD1
Item 3 >SHADD2
Item 4 >return
SI-Path >
```

The keyword length of -12 (12 characters or 6 words), minimum number of characters per keyword of 3, and average number of keywords of 4 are automatically applied to only the first SI-subkey (the other SI-subkeys are not keyworded).

Defining ASK Date SI-paths

Appending **/A** to the SI-path name defines an SI-path as an ASK Date Index and that the first item defined is an ASK Date:

```
SI-Path > ASKDATE/A
Item 1 >INVDAT
Include the Century (Y/n)? Y
Include the Year (Y/n)? Y
Include the Month (Y/n)? Y
Include the Day (Y/n)? Y
Use the New ASK Date format (Y/n)? Y
New ASK Date conversion. Format = CCYYMMDD
Item 2 >return
SI-Path >
```

After the path has been defined as an ASK Date SI-Path, SIMAINT will ask for information on the storage and conversion of the index. The first four questions refer to how the date should be stored, and therefore, how it will be retrieved. For example, if the date should only include the year and month, enter **N** for the century and day.

There are two date conversion routines for ASK. The "New ASK Date" format is valid for ASK Version 8.0 and later. If your version of ASK is earlier than 8.0, enter **N** for the New ASK Date format.

In this example, the data item, INVDAT, will be stored as an ASCII index with century, year, month and day. If an entry had a value of 8224 for INVDAT, the index value would be 19940506 (May 6, 1994).

Defining PowerHouse Date SI-paths

Appending **/P** to the SI-path name defines an SI-path as an PowerHouse Date Index and that the first item defined is a PowerHouse Date:

```
SI-Path > PHDATE/P
Item 1 >INVDAT
Include the Century (Y/n)? Y
Include the Year (Y/n)? Y
Include the Month (Y/n)? Y
Include the Day (Y/n)? Y
PowerHouse Date conversion.  Format = CCYYMMDD
Item 2 >return
SI-Path >
```

After the path has been defined as a PowerHouse Date SI-Path, SIMAINT will ask for information on the storage and conversion of the index. The four questions refer to how the date should be stored, and therefore, how it will be retrieved. For example, if the date should only include the year and month, enter **N** for the century and day.

Defining grouped SI-paths

A grouped SI-path is defined by specifying multiple SI-keys under a common SI-path name. It is not necessary to configure all members of the group at the same time: additional SI-keys may be added into the group at a later time (when deleting a grouped SI-path, however, all members are deleted).

These are the steps to follow in defining a grouped SI-path:

1. Specify a new SI-path name, optionally suffixed with **/B** (to index all blank SI-keys in the group) or **/K** (to define all SI-keys in the group as keyworded). The suffix attribute is inherited by all SI-keys in the group.
2. Specify the name of the first item to be contained as an SI-key in the group. Define the longest item first, since the SI-key length for the group may not exceed the length of the first item specified. If the group is related to a master dataset and includes the IMAGE search field, define it last.
3. Specify the SI-key length, which will be applied to all SI-keys in the group and which therefore must be no greater than the longest item in the group. SI-keys for items that are shorter than this length are padded with spaces.
4. Define additional SI-subkeys, if desired, when prompted. These SI-subkeys are unconditionally applied to all SI-keys in the group.
5. When prompted for the next SI-path name, specify the same SI-path name as before but append the suffix **/G**.
6. Enter the name of the next SI-key to be included in the group.
7. Repeat steps 5 and 6 for each additional SI-key in the group.

This example shows a grouping of two SI-key fields under the SI-path name *SHADD*:

```
SI-Path >SHADD
Item 1 >SHADD1
Item length is 13 words. Enter shorter length (- =Bytes) if desired >-10
Item 2 >return
SI-Path >SHADD/G
Item 1 >SHADD2
SI-Path >
```

❖ **The first field defined is automatically included in the group, even though /G was not appended to the corresponding SI-path name.**

❖ **The maximum number of items allowed in a group is 32.**

This example shows a grouping of three fields under the SI-path *CUPHN*.

```
SI-Path >CUPHN
Item 1 >CUPHN1
Item 2 >return
SI-Path >CUPHN/G
Item 1 >CUPHN2
SI-Path >CUPHN/G
Item 1 >CUPHN3
SI-Path >
```

The SI-key length was not prompted for here because all items are integers (data type I).

Defining super-grouped SI-paths

A super-grouped SI-path is defined by configuring an SI-path for a master dataset and then referencing that SI-path for one or more related detail datasets. It is not necessary to include all related detail datasets in the super-group.

The SI-set number option on the datasets, both master and details, must be the same if a super-group is wanted.

SI-path names between masters and their related details should be uniquely named to avoid any confusion between super-grouped paths, and those that are not super-grouped.

❖ **If the index is concatenated, the second and third items selected must exist in all datasets involved.**

These are the steps to follow in defining a super-grouped SI-path:

1. First configure a new SI-path for the master dataset, optionally suffixed with **/B** (to index all blank SI-keys in the group) or **/K** (to define all SI-keys in the group as keyworded). The suffix attribute is inherited by all SI-keys in the super-group.
2. Specify the name of the first item in the master set to be contained as an SI-key in the super-group. If no item in the master set is to be included in the super-group (or the master set is and automatic), specify the item from one of the related detail sets that will be part of the super-group.
3. Define additional SI-subkeys in the master set, if desired, when prompted. These SI-subkeys are unconditionally applied to all SI-keys in the super-group. If defining additional SI-subkeys for the master path, each subkey must also exist in the detail dataset(s), except for SI-subkey 1.
4. When prompted for the next SI-path name, press **RETURN** to be prompted for the next dataset (or continue to define additional, unrelated SI-paths for the master set).
5. When prompted for the next dataset, specify the name of a detail dataset related by an **IMAGE** path to the master set previously specified.
6. When prompted for SI-path name, enter the same SI-path name as for the master set but append the suffix **/G**.
7. Enter the name of the item in the detail dataset that is to be included as an SI-key in the super-group.
8. Enter **RETURN** for SI-path name (or continue to define additional, unrelated SI-paths for that detail set).
9. Repeat steps 5 through 9 for each additional related detail datasets in the super-group.

This example shows a super-grouping of a master dataset and a single associated detail set (related by the **IMAGE** path along **CUSTOMER-NUMBER**) under the SI-path name **CUSTOMER-BRANCH**:

```

Dataset >CUSTOMERS
SI-Path >CUSTOMER-BRANCH
Item 1 >CUSTOMER-NAME
Item length is 15 words. Enter new length (- =Bytes) if desired >return
Item 2 >return
SI-Path >return
Dataset >BRANCHES
SI-Path >CUSTOMER-BRANCH/G
Item 1 >BRANCH-NAME
SI-Path >return
Dataset >

```


This example shows a super-grouping of the ORDER-LINES detail dataset that is related to the ORDER-HEADERS master dataset in which the master dataset does not have any items included in the super-group, under the SI-path name *PART-DESCRIP*:

```
Dataset >ORDER-HEADERS
SI-Path >PART-DESCRIP
Item 1 >PART-DESCRIPTION      << does not exist in ORDER-HEADERS but in ORDER-LINES >>
Item length is 13 words. Enter new length (- =Bytes) if desired >return
Item 2 >return
SI-Path >return
Dataset >ORDER-LINES
SI-Path >PART-DESCRIP/G
Item 1 >PART-DESCRIPTION
SI-Path >return
Dataset >
```

Defining SI-paths that are both keyworded and grouped

As shown above, a grouped SI-path may be defined as keyworded, in which case every SI-key in the group is treated as keyworded.

The SI-path in this example is both keyworded and grouped, as defined simply by appending */K* to the SI-path name when first declared:

```
SI-Path >CUADD/K
Item 1 >CUADD1
Item length is 13 words. Enter keyword length >5
Enter minimum number of characters per keyword (1-4) >1
Enter average number of keywords per entry >4
Item 2 >return
SI-Path >CUADD/G
Item 1 >CUADD2
SI-Path >
```

Defining custom SI-paths

For custom SI-paths, for which the SI-key is generated by the SIUSER procedure, specify an arbitrary SI-path name and RETURN for the first SI-subkey:

```
Dataset >SHIP
SI-Path >CUSTOM-ID
Item 1 >return
Enter SI-Key length >4
Enter average number of indices per entry >4
SI-Path >
```

The *SI-key length* represents the length, **excluding** the SI-extension, that is returned in the *indices* parameter of the SIUSER procedure. The *average number of indices* represents the average value of the index count returned in the first word of the *index* parameter of SIUSER.

If the database is enabled for NLS (Native Language Support), the following question is also asked:

```
Use NLS-Sorting (N/Y) >
```

This determines whether NLS collating sequences are used for ordering entries for this SI-path. Because numeric data is sorted the same, respond **N** if the SI-path contains only numeric data fields; otherwise respond **Y**.

Defining independent SI-paths

For independent SI-paths that are not associated with any dataset, enter a **SPACE** and **RETURN** for the dataset and an arbitrary SI-path name:

```
Dataset >space + return
SI-Path >DOCUMENT-NAME
Enter SI-Index length >10
SI-Path >
```

❖ **The *SI-index length* represents the length of the entire SI-index: both the SI-key and SI-extension. This must be at least 2 words long.**

If the database is enabled for NLS, the following question is also asked:

```
Use NLS-Sorting (N/Y) >
```

This determines whether NLS collating sequences are used for ordering entries for this SI-path. Because numeric data is sorted the same, respond **N** if the SI-path contains only numeric data fields; otherwise respond **Y**.

Defining keyword exclusion SI-path

The special independent SI-path named **KWEXCLUDE** is reserved for words excluded from keywording (as defined in the file KWEXCLUD, which should already exist in the same group/account as the database). Enter **SPACE** for the dataset and **KWEXCLUDE** for the SI-path:

```
Dataset > space + return
SI-Path >KWEXCLUDE
Enter keyword length >5
SI-Path >
```

The *keyword length* represents the significant length of each exclusion word. Refer to the *Excluding words from keywording* chapter in this section for a discussion.

Deferring indexing

By default, SIMAINT will proceed to the extension phase (if required) and indexing phase once all configuration information has been specified, as indicated by a response of **RETURN** at the **DATASET>** prompt. You may instead want to defer indexing until a more convenient time, since it does require processing time.

To facilitate this, SIMAINT is capable of completing the extension phase (if required) and saving the newly-configured SI-definitions but deferring the indexing phase until explicitly specified. This is accomplished by responding **/N** to the **DATASET>** prompt, as shown:

```
Dataset>/N
```

To proceed with indexing when convenient, rerun SIMAINT against the same database and press **RETURN** at the **DATASET>** prompt.

Extension phase -- specifying SI-dataset(s) capacity

Once all definitions have been entered, SIMAINT proceeds to calculate the capacities for every SI-dataset. The calculated SI-dataset capacities are worst-case calculations which are sufficient for storing the SI-indices that are required based on the current capacities of the datasets they represent. (Refer to the *Internal Structures* appendix for more information about how the SI-dataset capacity is calculated.)

The capacities are displayed and the opportunity is given to override them. The program then proceeds to add the SI-item and SI-dataset(s) into the database.

```

Calculated capacity of SI-Dataset 241
Desired capacity (RETURN = Calculated capacity) >400

Calculated capacity of SI1-Dataset 160
Desired capacity (RETURN = Calculated capacity) >250
Extension started - Do not interrupt
Extension successful

```

The configuration process is now complete and the program proceeds to create the SI-indices.

Indexing phase -- progress display

Once the extension is completed, SIMAINT proceeds to generate the SI-path definitions and SI-indices. Extensive progress reporting is displayed during the indexing phase.

Each non-independent SI-path must go through three processes. First, the records in the dataset must be read, called the INPUT phase. Next the created indices for the SI-path must be sorted, called the SORT phase. Finally the indices must be written out to the SI dataset, called the OUTPUT phase.

During the INPUT and OUTPUT phases a display of the number of entries read, or indices added, the percentage complete, the elapsed time, and the elapsed CPU seconds are displayed at a regular predefined interval. During the SORT phase, no progress reporting can be displayed until after the SORT is completed.

```

Processing SI-Path KWEXCLUDE          OF
Processing SI-Path CUSTOMER-NAME      OF CUSTOMERS          # of Ent: 1002
    Input:      1002 records    100 %    CPU 0:00:01.2 Elapsed 0:00:02
    Sort:       1002 Indices          CPU 0:00:00.0 Elapsed 0:00:00
    Output:     1002 Indices    100 %    CPU 0:00:00.9 Elapsed 0:00:02
Processing SI-Path CUSTOMER-NAME-KW  OF CUSTOMERS          # of Ent: 1002
    Input:      1002 records    100 %    CPU 0:00:02.7 Elapsed 0:00:05
    Sort:       2997 Indices          CPU 0:00:00.0 Elapsed 0:00:00
    Output:     2983 Indices    100 %    CPU 0:00:01.6 Elapsed 0:00:03
Processing SI-Path ADDRESS1-CITY-KW  OF CUSTOMERS          # of Ent: 1002
    Input:      1002 records    100 %    CPU 0:00:04.3 Elapsed 0:00:07
    Sort:       4448 Indices          CPU 0:00:00.0 Elapsed 0:00:00
    Output:     4417 Indices    100 %    CPU 0:00:02.6 Elapsed 0:00:03

END OF PROGRAM

```

-
- ❖ Notice that the number of records read during the INPUT phase was the same as the **# OF ENT :**, but the number of indices sorted and written out either match the number of entries or is larger. This is because an SI-path that is not a one-to-one path (only simple paths or simple-concatenated paths are one-to-one paths) creates more indices than number of records.
-

- ❖ The number of indices written out may be less than the number sorted because all duplicate SI-indices and, unless otherwise requested, blank fields are not added. In other words, if a record has the same word in a keyworded SI-path only one SI-index will be added.
-

The default interval for updating the progress report is 1000 records or indices. This can be overridden by setting the JCW **SICOUNT**. The valid values for this JCW are 100 to 32767, inclusive. If the value is not valid, then 1000 will be used.

Example of configuring and establishing SI-indices for a database

A complete example of an entire SIMAINT configuration session for a database follows:

```

:RUN SIMAINT.SUPERDEX.SYS
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database > OEDB
Database has not been initialized for SUPERDEX
after the dialog phase, the program will go into privileged mode
to add the SI-Dataset(s) to the database
Dataset > space + RETURN
SI-Path > KWEXCLUDE
Enter keyword length > 4
SI-Path > RETURN
Dataset > CUSTOMERS
SI-Path > CUSTOMER-NAME
Item length is 15 words. Enter shorter length (- =Bytes) if desired)>RETURN
Item> RETURN
SI-Path> CUSTOMER-NAME-KW/K
Item 1 > CUSTOMER-NAME
Item length is 15 words. Enter keyword length > 4
Enter minimum number of characters per keyword >1
Enter average number of keywords per entry >6
Item 2 > RETURN
SI-Path > ADDRESS1-CITY-KW/K
Item 1 > ADDRESS-1
Item length is 13 words. Enter keyword length> 4
Enter minimum number of characters per keyword >1
Enter average number of keywords per entry >4
Item 2 > RETURN
SI-Path > ADDRESS1-CITY-KW/G
Item 1 > CITY
SI-Path > RETURN
Dataset > ORDER-HEADERS
SI-Path > CUSTOMER-NUMBER
Item 2 > RETURN
SI-Path > RETURN
Dataset > ORDER-LINES
SI-Path > ORDER-PART
Item 1 > ORDER-NUMBER
Item 2 > PART-NUMBER
Item length is 7 words. Enter shorter length (- =Bytes) if desired > RETURN
Item 3 > RETURN
SI-Path > PART-ORDER
Item 1 > PART-NUMBER
Item length is 7 words. Enter shorter length (- =Bytes) if desired > RETURN
Item 2 > ORDER-NUMBER
Item 3 > RETURN
SI-Path > RETURN
Dataset > RETURN

```

```
Calculated capacity of SI-Dataset 752
Desired capacity (RETURN = Calculated capacity) > RETURN
Extension started - Do not interrupt
Extension successful

Processing SI-Path KWEXCLUDE          of
Processing SI-Path CUSTOMER-NAME      of CUSTOMERS          # of Ent: 1003
  Input: 1003 Records 100 % CPU 0:00:01.3 Elapsed 0:00:02
  Sort: 1003 Indices CPU 0:00:00.0 Elapsed 0:00:00
  Output: 1003 Indices 100 % CPU 0:00:00.9 Elapsed 0:00:01
Processing SI-Path CUSTOMER-NAME-KW  of CUSTOMERS          # of Ent: 1003
  Input: 1003 Records 100 % CPU 0:00:03.0 Elapsed 0:00:03
  Sort: 3046 Indices CPU 0:00:00.0 Elapsed 0:00:00
  Output: 3032 Indices 100 % CPU 0:00:01.8 Elapsed 0:00:02
Processing SI-Path ADDRESS1-CITY-KW  of CUSTOMERS          # of Ent: 1003
  Input: 1003 Records 100 % CPU 0:00:04.6 Elapsed 0:00:05
  Sort: 4375 Indices CPU 0:00:00.0 Elapsed 0:00:00
  Output: 4344 Indices 100 % CPU 0:00:02.6 Elapsed 0:00:03
Processing SI-Path CUSTOMER-NUMBER   of ORDER-HEADERS      # of Ent: 2620
  Input: 2620 Records 100 % CPU 0:00:02.4 Elapsed 0:00:03
  Sort: 2620 Indices CPU 0:00:00.0 Elapsed 0:00:00
  Output: 2620 Indices 100 % CPU 0:00:01.4 Elapsed 0:00:02
Processing SI-Path ORDER-PART        of ORDER-LINES      # of Ent: 9272
  Input: 9272 Records 100 % CPU 0:00:09.1 Elapsed 0:00:10
  Sort: 9272 Indices CPU 0:00:00.1 Elapsed 0:00:00
  Output: 9272 Indices 100 % CPU 0:00:06.9 Elapsed 0:00:09
Processing SI-Path PART-ORDER        of ORDER-LINES      # of Ent: 9272
  Input: 9272 Records 100 % CPU 0:00:09.1 Elapsed 0:00:10
  Sort: 9272 Indices CPU 0:00:00.1 Elapsed 0:00:00
  Output: 9272 Indices 100 % CPU 0:00:07.3 Elapsed 0:00:09
Total time : CPU 0:00:57.8 Elapsed 0:06:27

END OF PROGRAM
```

Running SIMAINT in batch

SIMAINT can be run in batch, and uses dialog similar to on-line. The method for creating a job stream by which to run SIMAINT in batch is to build a job file using the SCHEMA entry-point after all paths have been defined.

The discrepancies between on-line and batch use are:

- all prompts are displayed during batch, while during on-line the prompts are variable depending on answers
- a line containing only a **SPACE** is represented in batch by a blank line
- a line containing only a **RETURN** (which is normally specified in a batch job as a blank line) is represented by a line containing a double slash (//) in the first two character positions

SIMAINT will QUIT (not TERMINATE) normally upon encountering any error in batch, permitting testing of the system JCW.

Running SIMAINT with a **;STDIN** option will cause SIMAINT to execute as if in batch.

SECTION 4:

Programming

Overview

This section discusses various methods utilized in programming with SUPERDEX and gives examples of the different types of SUPERDEX access.

Although the SUPERDEX intrinsics are discussed throughout this section, they are documented fully for reference in the *Intrinsics* section.

Chapter 1 Adding, updating, and deleting entries

Function discusses *Adding, updating, and deleting entries* using DBPUT, DBUPDATE, and DBDELETE. Also covered are custom indices, the DBPUTIX and DBDELIX intrinsics, and independent SI-paths.

Chapter 2 Qualifying entries with DBFIND

Function describes various methods of *Qualifying entries with DBFIND*. All the various access methods are shown, including keyworded and grouped retrieval, as well as lookups that involve a single set, multiple SI-paths in a single set, multiple sets, and multiple bases.

Chapter 3 Retrieving entries with DBGET

Function looks at *Retrieving entries with DBGET* in sorted sequential order using *modes 5, 6, 15, and 16*, as well as methods for reading masters and their related details and SI-indices only.

Chapter 4 Additional programming considerations

Function examines *Additional programming considerations*, such as programming language variations.

Chapter 5 Native Language Support

Function describes special considerations that must be made when using HP's *Native Language Support* facility, for adding, updating, and indexing entries and qualifying entries with DBFIND.

CHAPTER 1:

Adding, updating, and deleting entries

Adding and deleting entries with DBPUT and DBDELETE

Entries are added and deleted using TurboIMAGE's DBPUT and DBDELETE intrinsics, which additionally maintain the associated SI-indices.

Determining SI-key value

SUPERDEX automatically determines, based on its configuration, the SI-key and SI-subkey values to include in the SI-indices.

There are instances, however, in which the SI-key value is not represented in the data in any straightforward manner and SUPERDEX is therefore unable to determine it. Some examples of this are as follows:

- **Date reformatting**
Date must be converted from *yy/mm/dd* format into Julian or other format
- **Upshifting**
SUPERDEX only upshifts SI-keys for SI-paths that are configured as keyworded. You may want to upshift SI-keys for non-keyworded SI-paths
- **Compound indexing**
SUPERDEX allows a maximum of four SI-subkeys in an SI-key, which may be insufficient

SUPERDEX provides two different facilities for addressing these needs: the **SIUSER** procedure, and the **DBPUTIX** and **DBDELIX** intrinsics.

Custom SI-indices with SIUSER

SIUSER is a user-written procedure that is used by SUPERDEX to compute one or more custom SI-indices for entries whenever DBPUT, DBUPDATE, or DBDELETE is called and whenever SIMAINT is run.

SIUSER requires that the database, dataset, and SI-path be defined in its *base*, *dset*, and *item* parameters and that the full data from the record be supplied in its *buffer* parameter. SIUSER returns the number of SI-indices created and their values in the *index* parameter.

Explicit SI-index management with DBPUTIX and DBDELIX

The DBPUTIX and DBDELIX intrinsics explicitly add and delete SI-indices, and are useful for generating custom SI-indices in addition to those automatically generated by DBPUT and removed by DBDELETE.

The common method for establishing multiple SI-indices for an entry, over and above those generated by DBPUT, is to first call DBPUT to add the entry to the dataset and create the configured SI-indices, and then call DBPUTIX as many times as necessary to establish the additional SI-indices. The same technique is used with DBDELETE and DBDELIX to remove SI-indices.

DBPUTIX and DBDELIX require that the database, dataset, and SI-path be defined in their *base*, *dset*, and *item* parameters and that the full SI-index, including the SI-extension, be supplied in the *buffer* parameter.

For a master set, the SI-extension is the entry's IMAGE search field value, which for DBPUT may be retrieved from the *buffer* parameter and for DBDELETE may be gotten from the *list* used by the DBGET that located the entry for deletion. For a detail set, the SI-extension is the entry's relative record number, which is returned in words 3-4 of the *status* array from the DBPUT or DBDELETE.

Independent SI-paths

The DBPUTIX and DBDELIX intrinsics are also used for independent indexing, in which the entities being indexed do not reside in IMAGE datasets.

DBPUTIX and DBDELIX against independent SI-paths require that the *base* parameter identify the database in which the SI-indices are contained and the *dset* parameter be left blank or set to **200**. The SI-path is defined in the *item* parameter, and the full SI-index, including an appropriate SI-extension, is provided in the *buffer* parameter.

CHAPTER 2:

Qualifying entries with DBFIND

In IMAGE, entries in a detail dataset are normally accessed by using DBFIND to locate the head of a chain (which is contained in a related master dataset) for a specified search field value (commonly called a "key value"), and then the associated entries are subsequently retrieved via DBGET *mode 5* or *6*. An entry in a master dataset is normally accessed via DBGET *mode 7*, based on the search field value specified.

SUPERDEX also uses DBFIND and DBGET to access dataset entries. The same techniques are used for both master and detail datasets in the same way.

In SUPERDEX, DBFIND does much more than locate the chain head for a specified key value: it qualifies multiple entries based on various criteria, and may be called multiple times in succession to refine the selection using various operators, fields, datasets, and databases.

Once entries have been qualified with DBFIND, they may be retrieved in ascending or descending sorted sequential order with DBGET *modes 5* and *6* or DBGET *modes 15* and *16*.

Qualifying and retrieving entries in a master or detail dataset using an SI-path is just like qualifying and retrieving entries in a detail set using an IMAGE path.

Summary of DBFIND options

Whereas IMAGE's DBFIND works only on detail datasets, SUPERDEX's DBFIND works on:

- detail datasets
- master datasets
- SI-indices only, which may reference external files

Based on how SI-paths are configured, DBFIND can qualify entries by any of the following:

- any word in a keyworded SI-key
- a combination of SI-subkey values in a concatenated SI-key
- a value that occurs in multiple SI-keys that are grouped
- any subitem in a compound IMAGE item that is configured as an SI-key

DBFIND can qualify entries generically by any of the following, as specified in the *argument* by conditional and relational operators:

- start with a specified value (partial key)
- contain a specified value (embedded key)
- greater-than or equal-to a specified value
- less-than or equal-to a specified value
- not equal to a specified value
- in a range of two values

DBFIND also permits multiple generic or exactly-matching values to be specified in an *argument*, which permits the use of range and Boolean operations to find all qualifying entries:

- by an **AND** combination of multiple values
- by an **OR** combination of multiple values
- by an **AND NOT** combination of multiple values

In performing *relational access*, multiple successive DBFINDs may be called and their results ANDed, ORed, and AND NOTed by Boolean operators. This dynamically achieves the following operations:

- refine a selection with additional criteria
- undo a selection (revert to results of prior DBFIND)
- qualify by multiple SI-keys in a single dataset
- qualify by multiple datasets in a database
- qualify by multiple databases

Indexed access vs. relational access

Internally, one of two access methods is used in qualifying entries with DBFIND: *indexed access* or *relational access*.

With indexed accessed, DBFIND locates the first qualifying SI-index that matches the specified *argument* in the SI-path defined by the *item* parameter. The B-tree in the corresponding SI-dataset is accessed, and all qualifying entries form a logical SI-chain. Entries on the SI-chain are returned using DBGET by reading up or down the B-tree and retrieving the corresponding entries one-by-one.

With relational access, SI-indices are qualified using the same method but are then copied to form a virtual SI-chain in the *active SI-subset*. Entries are retrieved using DBGET by reading up or down this virtual SI-chain. The advantage is that unlike indexed access, the SI-chains contained in the active SI-subset are available for combination with entries located by subsequent DBFIND calls.

To recap, *indexed access* retrieves entries by directly reading a logical SI-chain in the SI-dataset; *relational access* first forms the logical SI-chain and then copies it to form a virtual SI-chain in the active SI-subset.

Indexed access is used for retrievals that can be accomplished by accessing a single SI-chain. This accounts for most retrievals, and is used by default.

Relational access is used for Boolean retrievals that require the use of multiple SI-chains, such as performing retrievals against multiple SI-paths, sets, and bases by using multiple DBFIND calls. It is invoked when calling DBFIND with *mode* 12, or DBFIND *mode* 1 explicitly specified by enclosing the complete *argument* with tilde and semi-colon (*~ . . . ;*).

DBFIND modes

Typically, most DBFINDs are performed using either *mode 1* or *mode 21*. Either *mode* may be used for both indexed and relational access methods, although they are treated identically when performing relational access. In indexed access, both *modes* perform the same function, except *mode 21* does not return the qualifying number of entries in the *status* array (returns a count of 1 if there are qualifying entries) and may therefore be considerably more efficient than *mode 1*. *Mode 21* should be used instead of *mode 1* in indexed access whenever possible for efficiency.

For relational access, DBFIND *mode 12* can be used instead of *mode 1*. This mode forces a relational DBFIND and allows for the Boolean operators. It is also used in conjunction with DBFIND *mode 13*. When DBFIND *mode 12* is called, a relational list of the qualified records is created. If a previous relational list exists, this list is saved. DBFIND *mode 13* will replace the current relational list with the previous relational list.

Modes 1nn, 2nn, 3nn, 4nn, and 5nn in indexed access read *nn* words of the *argument* and set the SI-pointer before or after the first qualifying entry, depending on the mode. Additionally, a working chain of all the qualified data is enforced. If *nn* is prefixed by a minus sign (-), they read *nn* bytes instead of words. These *modes* are refinements and therefore do not handle as many *argument* constructs as *modes 1, 12 and 21*, but are useful for certain circumstances, such as approximate match retrieval in which no qualifying value exists which sets the SI-pointer to the nearest qualifying entry. These *modes* are also useful in the unlikely event that a combination of symbols used to represent a SUPERDEX operator (e.g. >=) conflicts with a value in an SI-key in a data entry. Also, if mixed data types or numeric data types are used within a concatenated index, these modes can be utilized to qualify records based on more than just the first item in the index.

Modes 1nn and *5nn* may also be used in relational access against a virtual SI-chain in the active SI-subset if a null *item* is specified. Refer to *Positioning on virtual SI-chain* later in this section for a discussion.

Mode 100 positions the SI-pointer at the logical beginning of the dataset (lowest entry in ascending sorted sequential order) or the beginning of a virtual SI-chain in the active SI-subset. *Mode 500* positions the SI-pointer at the logical end of the dataset (highest entry in descending sorted sequential order) or the end of a virtual SI-chain in the active SI-subset.

DBFIND arguments used for indexed access

For indexed access, if DBFIND is called in *mode 1* or *21* and the specified *argument* value is not the full SI-key value, either the *buffer* must be padded with spaces or the *argument* value must be terminated by:

- for alphanumeric fields, a single blank followed by a single @
- for alphanumeric fields, a single @ (used for partial-key retrieval)
- for numeric fields, a single blank

For DBFIND *modes 1* and *21* in indexed access, ASCII numbers may be specified for most numeric items (data types I, J, K, P, R, E, and Z) if prefixed with ==, >=, <=, or <> or, if appropriate, a - (negative sign). The == operator simply converts an ASCII value specified to binary format for comparison.

DBFIND *mode 11* or *22* (binary range modes) can also be used to qualify entries based on binary values. It is important to remember that the range must be entered in the *argument*. If only one value is to be qualified, the start and stop values are the same. For example, if the selected records should have values of less than 0, the first value of *argument* must be the lowest possible value for the data type, not 0. If the first value in the *argument* is 0, then no negative values will qualify.

❖ For concatenated SI-keys that contain SI-subkeys of mixed data types (alphanumeric vs. numeric) and for which the first SI-subkey is numeric, an ASCII value may be specified in *modes* 1 and 21 for the first SI-subkey only.

❖ When performing relational access against numeric items, the *argument* must be specified in ASCII.

For data types P and Z, SUPERDEX's DBFIND treats unsigned and positive values equivalently.

Real numbers (items of data type R or E) may include embedded decimal points (**.**), exponential signs (**E**), and positive (**+**) and negative (**-**) signs.

DBFIND mode/argument examples

Simple SI-key: alphanumeric X12		
<i>mode</i>	<i>argument</i>	<i>description</i>
1	GOLDENBERG	exact match on "GOLDENBERG"
21	GOLDENBERG	same, but does not return the number of qualifying entries in the <i>status</i> array
102		
1	GOLD@	all that start with "GOLD"
1	GOLD	same
1	G?LD	the values "GELD" and "GOLD"
1	G?LD@	the values "GELD," "GOLD," "GOLDEN," and "GOLDBERG"
	?OLD	
1	PX??4400@	the values "BOLD," "COLD," "FOLD," and "GOLD"
1		all that start with "PX," followed by any two characters,
100	PX??44@	followed by 4400, followed by anything
500		all with "PX" followed by any two characters, followed by "44,"
	>=A@<=B@	followed by anything
	<>87@	all in the range between "A" and "B," inclusive
	(ignored)	all except those that begin with "87"
	(ignored)	the first alphabetical value in the dataset on the specified SI-path
		the last alphabetical value in the dataset on the specified SI-path

Simple SI-key: numeric R2		
<i>mode</i>	<i>argument</i>	<i>description</i>
1	>=1000	greater than or equal to 1000
1	>=123.4E5	greater than or equal to the specified number
1	<=100	less than or equal to 100
1	>=100<=1000	all in the range between 100 and 1000, inclusive

Simple SI-key: numeric Z4		
<i>mode</i>	<i>argument</i>	<i>description</i>
1	1234	exact match on the value 1234
102	1234	same
-103	123	exact match on 123
any	0034	the value 34 (leading zeroes must be specified)
any	000J	the signed value -1 (leading zeroes must be specified)
1	>=1234	all greater than or equal to 1234
1	<=000J	all less than or equal to the signed value -1
1	>=23<=24	all in the range between 23 and 24 inclusive (leading zeroes need not be specified if prefixed with ==, >=, <=, or <>)

Concatenated SI-key: X4 + X4		
<i>mode</i>	<i>argument</i>	<i>description</i>
1	ABCD@	"ABCD" in the first SI-subkey, any value in the second SI-subkey
1	ABCD1234	exact match on both SI-subkeys: "ABCD" and "1234"
104	ABCD1234	same
-108	ABCD1234	same
1	????1234	"1234" in the second SI-subkey, any value in the first SI-subkey
21	????1234	same, but does not return the number of qualifying entries in the <i>status</i> array
-111	ABCD123456	condition word -31 ("BAD MODE"), because the SI-key length was exceeded

Concatenated SI-key: X4 + I1 (values underlined are specified in binary)		
<i>mode</i>	<i>argument</i>	<i>description</i>
-106	ABCD <u>1234</u>	exact match on "ABCD" in the first SI-subkey and 1234 in the second SI-subkey (where 1234 is specified in binary)
-105	ABCD1234	condition word -31 ("BAD MODE"), because I1 requires 2 bytes and therefore <i>mode</i> -106 should be used
1	ABCDE	all with "ABCD" in the first SI-subkey, and any value in the second SI-subkey (the extra character is ignored)
21	ABCDE	same, but does not return the qualifying number of entries in the <i>status</i> array

Concatenated SI-key: I1 + X4 (values underlined are specified in binary)		
<i>mode</i>	<i>argument</i>	<i>description</i>
103	<u>1234</u> ABCD	exact match on the value 1234 in the first SI-subkey and "ABCD" in the second SI-subkey (where 1234 is specified in binary)
-103	<u>1234</u> A	all with 1234 in the first SI-subkey and "A" leading the second SI-subkey (truncated value)
1	<u>1234</u> A	all with 1234 in the first SI-subkey (the extra character is ignored)
1	>=1234	all greater than or equal to 1234
1	<=2999	all less than or equal to 2999
1	>=-100<=100	all greater than or equal to -100 (negative 100) and less than or equal to 100 (positive 100)
-101	spaces	condition word -31 ("BAD MODE"), because the <i>mode</i> does not match the argument length
1	1234	all equal to 12624 (first word, value "12", converted from ASCII value to binary value)

Concatenated SI-Key: Z4 + X4		
<i>mode</i>	<i>argument</i>	<i>description</i>
104	1234ABCD	exact match on the value 1234 in the first SI-subkey and "ABCD" in the second SI-subkey
-106	1234AB	exact match on 1234 in the first SI-subkey and "AB" in the second SI-subkey
1	1234	all with 1234 as the first SI-subkey
1	1234ABCD	same (the value for the second SI-subkey is ignored)
any	0034	the value 34 (leading zeroes must be specified)
1	0034ABCD	same (the value for the second SI-subkey is ignored)
1	>=1234	all entries with first SI-subkey greater than or equal to 1234
1	>=1234<=2345	all entries with first SI-subkey between 1234 and 2345 inclusive
1	>=1234ABCD	results unpredictable; any characters specified after the first SI-subkey renders the entire argument invalid
1	>=1234AB<=1234AB	same

Finding entries using a partial key

DBFIND can search for entries using a partial key value, with the @ character. The @ is treated as a wildcard (as in :LISTF) that stands for any number of any characters.

For example, to find all entries that begin with "GENERAL":

```
argument = GENERAL@
```

The partial key value specified is compared with the entries.

The @ can be used up to two times within an *argument*. This is accomplished by surrounding the requested *argument* with << >>. The format is <<A@B@C>>. This means that this argument will qualify records that begin with an A, having a B anywhere in the middle, and ends with a C. If only one @ is provided, there is an implied @ at the end of the argument (e.g.<<A@B>> is identical to <<A@B@>>).

For example, to find all entries that begin with "GE" and contain "AL":

```
argument = <<GE@AL>>
```



The @ can also be used as the first character in an argument. For example, <<@OWN>> will return BROWN and CROWN.

Finding entries using a generic key

The ? and # characters facilitate generic searches. The ? represents a single alphanumeric character and the # a single numeric character. They may occur multiple times anywhere in the value, for example:

```
argument = STR?NG
```

would locate "STRING," "STRONG," and "STRUNG."

```
argument = AP#J3@
```

would locate "AP1J379C," "AP8J3AQ4," and "AP4J3", but "APBJ3826" would not qualify.

Both matchcodes may be used in combination with each other, along with the @ wildcard, for example:

```
argument = AP?J#@
```

would additionally find "APBJ3826," "AP7J8AH," and "APZJ277." In this example, the @ acts as the terminator because the argument is not surrounded with << >>.

The matchcodes may also be used to locate entries in which the desired value does not begin in the first position; for example:

argument = **??RT?N**

would locate "BARTON," "BURTON," "MARTIN," and "MORTON."

This technique is especially useful for generic searches on concatenated SI-subkeys, by specifying wildcard conditional operators for the unspecified SI-subkeys. These three examples perform generic searches on the first, second, and third SI-subkeys, respectively, of a concatenated SI-key consisting of an X2, X4, and X4 field with the value "PH1234ABCD":

argument = **PH@**

argument = **??1234@**

argument = **?????ABCD**

❖ Instead of specifying a **?** at the beginning of an argument, it is more efficient to define an offset.

❖ For indexing purposes, a space is considered a valid character (which is different from :LISTF). For example, a code value may be "AP 3" or "AP3 ". Therefore, an argument value of "AP??" would qualify both code values.

Finding entries greater than or equal to a specified value

Greater-than-or-equal-to searches are accomplished using the **>=** operator to prefix the value. For example, to find all entries greater than or equal to 1000:

argument = **>=1000**

SUPERDEX does not have a greater-than operator: **>** is not recognized and is therefore treated as a regular character. To accomplish a greater-than search, add one to the value being searched for:

argument = **>=1001**

If the field being searched is of IMAGE data type R, specify the value in the following format:

argument = **>=1000<>1000**

Finding entries less-than or equal-to a specified value

Less-than-or-equal-to searches are accomplished using the `<=` operator to prefix the value. For example, to find all entries less than or equal to 500:

argument = `<=500`

SUPERDEX does not have a less-than operator: `<` is not recognized and is therefore treated as a regular character. To accomplish a less-than search, subtract one from the value being searched for:

argument = `<=499`

If the field being searched is of IMAGE data type R, specify the value in the following format:

argument = `<=1000<>1000`

Finding entries not equal to a specified value

To find entries not equal to a particular value, use the `<>` operator. For example, to find all unpaid orders:

argument = `<>PAID`

The `<>` operator may also be embedded within an *argument* to perform a Boolean AND NOT retrieval. For example, to find all the entries with ZIP-CODES (an X6 item) beginning with "900" but not in "90039":

argument = `900@<>90039`

Finding entries in a range of values

The `>=` and `<=` operators may be used in combination to specify a range. For example, to find all the entries that start with letters between "A" and "D," inclusive:

argument = `>=A@<=D@`

Pattern-matching and/or exclusion may optionally be performed within a range, allowing entries to be qualified that not only fall between two values but also conform to a specific pattern. For example:

argument = `#####AA>=8910@<=8912@`

would find entries that fall between the values "8910" and "8912" and additionally contain "AA" as the fifth and sixth characters.

◆ The argument following `>=`, `<=` and `<>` may NOT contain embedded `?` or `#` characters

Finding entries in a concatenated SI-key

In searching for entries using a concatenated SI-key where the data types of the SI-subkeys are the same, the entire concatenated value is considered. This permits an entry to be located by very specific criteria. For example, a concatenated SI-key comprised of an X2, X4, and X4 and containing the three SI-subkey values "PH," "1234," and "ABCD" would be located by:

argument = PH1234ABCD

If the concatenated SI-key is comprised of SI-subkeys that are all alphanumeric (data types X or U, as in the previous example), DBFIND *modes* 1, 12, 21, and 23 may be used to qualify entries using a full or partial SI-key value.

❖ **If, however, the SI-subkeys are of both alphanumeric and numeric data types, *modes* 1nn, 2nn, 3nn, 4nn, or 5nn must be used.** An exception is that if the leftmost SI-subkey is alphanumeric, a partial value appended with an @ may be specified to match on the leftmost alphanumeric SI-subkey(s). Also, wild cards and matchcodes cannot be used on numeric fields.

If the length of one of the SI-subkeys was specified with an odd number of bytes, a "?" or " " must be used to hold a place in the argument for concatenated SI-keys. For example, if the first SI-subkey was defined with a length of three bytes, and the second SI-subkey with four, and argument would look like:

argument = ABC?1234
or *argument* = ABC 1234

Finding entries in a group

Searching for entries in multiple SI-keys in a dataset that are grouped together is completely transparent--DBFIND treats an SI-path containing a group of SI-keys as if it were a single SI-key and unconditionally searches all SI-keys in the group.

For example, if the items PHONE-1, PHONE-2, and PHONE-3 were grouped together in an SI-path, a phone number contained in any of the three fields would be searched for in a single DBFIND call.

For searches against an SI-key in a group which is shorter than the other SI-keys in the same group and is therefore padded with spaces, it is necessary to either pad the *argument* with spaces or perform a partial-key retrieval when calling DBFIND.

Finding entries in a super-group

Searching for entries with multiple SI-keys in multiple sets that are super-grouped together qualifies master entries based on the contents of the SI-keys that form the super-group.

For example, if the book title is contained in the master dataset BOOK, the book author(s) is contained in a related detail set AUTHOR, and the book summary is contained in the related detail set SUMMARY, and the three sets are super-grouped together, a specified value would be searched for in all three datasets. Master entries that contain the specified value in the title would qualify, as well as master entries that contain the specified word in either related detail dataset.

To qualify master entries, DBFIND is called against the master set specifying the SI-path name of the super-group in the item parameter, as shown:

```
dataset = BOOK
SI-path = BOOK-KEY
mode = 1
argument = PLAN@
```

Finding entries in a compound IMAGE item

SI-keys that contain compound IMAGE items are automatically handled as if they were grouped; all subitems are always searched when the SI-path is referenced in the *item* parameter.

Finding entries by keyword

Searching for the occurrence of a keyword in an SI-path that has been configured as keyworded is completely transparent: the keyword is simply specified in the *argument* parameter.

If a given keyword occurs more than once in a single data entry, the entry is always returned only once. However, if multiple words in a single entry meet the search criteria, the entry will by default be returned multiple times. For example,

```
argument = PLAN@
```

an entry containing the value "THE PLANNING COMMISSION'S NEW PLAN" would be returned twice. To prevent entries from being returned multiple times, specify relational access by using *mode* 12 and placing a semi-colon (;) at the end of the "argument".

```
mode = 12
argument = PLAN@;
```

❖ It is possible that entries with keywords that exceed the configured keyword length will be erroneously qualified. For example, a keyworded SI-key with a configured keyword length of 4 words (8 characters) containing the value "INDUSTRIOUS" would be qualified by an *argument* of "INDUSTRIAL," since only 8 characters are indexed and matched on. If the length of a keyword specified exceeds the keyword length configured for the referenced SI-path, the argument value is truncated to the configured keyword length and matching is done based on the truncated value.

DBFIND arguments used for relational access

For relational access, any of the previous *arguments* used for indexed access may be specified, and operate in the same way.

Unlike indexed access, however, search values specified in ASCII for retrieval against a binary field are automatically converted (and therefore do not need to be prefixed by the `==` conversion operator).

There are three ways to specify the *argument* for relational access: the SQL Notation, the Infix Notation, and the Reverse Polish Notation (RPN). If using either SQL or Infix, there are also two modes. DBFIND *mode* 1 will allow relational access if the first position of the *argument* is equal to a tilde (~), or *mode* 12 will always enforce relation access. With either *mode* the *argument* **must** contain a semi-colon (;) in the last position. The advantage of *mode* 1 is that the user can easily specify whether they want relational access or indexed access (which is a faster process). The advantage of *mode* 12 is that the user does not need to know when they will need relational access and can always enter the *arguments* the same.

❖ The SQL Notation is the use of "AND", "OR", and "NOT" as Boolean operators to specify the relationship between the *argument* values. The operators can be in either upper- or lower-case, including any combination of both. The *argument* must be left justified and contain no embedded spaces within each *operand* value unless the *operand* value is enclosed in quotes. For example, an *argument* of `JONES @ OR SMITH@;` is not a valid argument because there is a space between the `S` and the `@` in the selection value `JONES@`.

SQL Notation Operators		
<i>Oper</i>	<i>Description</i>	<i>Format</i>
AND	AND'ed	x AND y;
OR	OR'ed	x OR y;
NOT	AND NOT'ed	x NOT y;

❖ Infix Notation is the use of "+", ",", and "-" as Boolean operators. There must not be any spaces within the entire *argument*. For example, the *argument* `JONES@, SMITH@;` is not valid. It must be `JONES@,SMITH@;`.

Infix Notation Operators		
<i>Oper</i>	<i>Description</i>	<i>Format</i>
+	AND'ed	x+y;
,	OR'ed	x,y;
-	AND NOT'ed	x-y;

❖ When Reverse Polish Notation (RPN) is used, each *argument* must be enclosed within square brackets ([]), and the entire *argument* must be terminated by a blank or @ (a value within square brackets need no trailing character).

Reverse Polish Notation Operators		
<i>Oper</i>	<i>Description</i>	<i>Format</i>
&	AND'ed	[x][y]&
	OR'ed	[x][y]
!&	AND NOT'ed	[x][y]!&

DBFIND mode/argument examples

Relational retrievals		
<i>mode</i>	<i>argument</i>	<i>description</i>
1	~GOLD@;	all that start with "GOLD"
12	GOLD@;	same
1	~>=A@ and <=B@;	all in the range between "A" and "B", inclusive
12	>=A@ AND <=B@;	same
12	>=A@+<=B@;	same
1	~A@ or B@;	all that begin with "A" or "B"
12	A@ OR B@;	same
12	A@,B@;	same
1	~>=A@ not AB@;	all greater than or equal to those that begin with "A", except those that begin with "AB"
12	>=A@ NOT AB@;	same
12	>=A@-AB@;	same

Finding entries by ANDing multiple values

A DBFIND *argument* may contain multiple values that are ANDed together with one of the AND Boolean operators (**AND**, **+**), and only those entries that qualify based on all specified values are selected.

For example, find all the entries in a grouped SI-path that contain both the values "JOHN" and "CHICAGO" using DBFIND *mode* 12:

```
argument = CHICAGO AND JOHN;      SQL Notation
argument = CHICAGO+JOHN;          Infix Notation
```

❖ In the above example "CHICAGO" was specified first and "JOHN" second. This was done because "CHICAGO" appears on fewer entries than "JOHN". It is faster and more efficient to specify the less common value first when performing relational access retrievals.

Finding entries by ORing multiple values

Multiple values in a DBFIND *argument* may be ORed together using one of the OR Boolean operators (**OR**, **,**), and entries that qualify based on any specified value are selected.

For example, find all the entries in a keyworded SI-path that contain the word "FITTING," "NIBBLE," or "CONNECTOR", again with DBFIND *mode* 12:

```
argument = FITTING OR NIBBLE OR CONNECTOR;      SQL Notation
argument = FITTING,NIBBLE,CONNECTOR;           Infix Notation
```

Finding entries by AND NOTing multiple values

Multiple values in a DBFIND *argument* may be AND NOTed together using one of the AND NOT Boolean operators (**NOT**, **-**), and entries that qualify based on one value and not another are selected.

For example, find all the entries in a non-keyworded SI-path that begin with "NEW" except those that begin with the value "NEW YORK", with DBFIND *mode* 12:

```
argument = NEW@ NOT "NEW YORK";      SQL Notation
argument = NEW@-"NEW YORK";          Infix Notation
```

❖ Incidentally, the same retrieval could also be performed more efficiently using indexed access with *mode* 1 DBFIND or *mode* 21:

```
argument = NEW@<>"NEW YORK"
```

Finding entries with combined Boolean operators

The Boolean operators used by SUPERDEX allow very powerful combinations of operations to be specified.

For example, DBFIND *mode* 12 can be used to find all the entries in a keyworded SI-key that contain both the words "COMB" and "BIND" or "HOLD" and "DRILL" but not any word beginning with "FASTEN":

argument = (COMB and BIND) or (HOLE and DRILL) not FASTEN@;
SQL Notation

argument = (COMB+BIND) , (HOLE+DRILL) -FASTEN@;
Infix Notation

Active and backup SI-subsets

Besides the active SI-subset in which the virtual SI-chain (selected by an *argument* using one of the Boolean operators) is copied, a *backup SI-subset* is used when processing *arguments* that contain more than one value.

The internal processing of the *argument* specified in the previous example is done as follows:

COMB The SI-chain that is formed by the selection is stored in the active SI-subset. Before this is done, the contents of the backup SI-subset (if not empty) is transferred to the 3rd level SI-subset, and the contents of the active SI-subset (if not empty) is transferred to the backup SI-subset.

and BIND A Boolean ANDing is performed between the result of this selection and the virtual SI-chain contained in the active SI-subset.

HOLE The virtual SI-chain contained in the backup SI-subset is transferred to the 3rd level SI-subset. Then the virtual SI-chain contained in the active SI-subset is transferred to the backup SI-subset, and the resulting SI-chain is stored in the active SI-subset (replacing the existing SI-chain).

and DRILL An ANDing is performed in the same manner as above.

or The contents of the 3rd level SI-subset are deleted, and then replaced by the backup SI-subset. Next, the contents of the backup SI-subset and the active SI-subset are ORed, and the resulting SI-chain is stored in the active SI-subset.

not FASTEN@ The negated result of the selection is ANDed with the SI-chain in the active SI-subset, with the resulting SI-chain stored in the active SI-subset, replacing the existing SI-chain.

At the end of the complete operation, the backup SI-subset will contain whatever SI-subset was contained in the original active SI-subset.

There are special operations that can be executed against the active and backup SI-subsets. **These operations require special operators using Reverse Polish Notation (RPN) and DBFIND mode 1.** These arguments should only be used with DBFIND mode 1 and should not be utilized with DBFIND mode 12 and DBFIND mode 13. The processes should not be mixed, and if so, can cause unexpected results. The effect of the various operations on the active and backup SI-subsets is summarized in the following table.

<i>argument</i>	<i>SI-chain in active SI-subset</i>	<i>SI-chain in backup SI-subset</i>	<i>SI-chain in 3rd level SI-subset</i>
&	ANDed with backup	replaced by 3rd level	deleted
 	ORed with backup	replaced by 3rd level	deleted
!&	AND NOTed with backup	replaced by 3rd level	deleted
[x]	replaced by <i>x</i>	replaced by old active	replaced by old backup
[x]&	ANDed with <i>x</i>	unchanged	unchanged
[x] 	ORed with <i>x</i>	unchanged	unchanged
[x]!&	AND NOTed with <i>x</i>	unchanged	unchanged
[x]\$	ORed with active after <i>x</i> is ANDed with backup(*)	unchanged	unchanged
/	unchanged	replaced by active	replaced by backup
[]	erased(*)	replaced by active	replaced by backup
\	replaced by backup	replaced by 3rd level	deleted
/\	unchanged	unchanged	deleted
\\ /	replaced by backup	unchanged	unchanged
^	swapped with backup	swapped with active	unchanged
\[x]	replaced by <i>x</i>	unchanged	unchanged
/[x]&	ANDed with <i>x</i>	replaced by old active	replaced by old backup
/[x] 	ORed with <i>x</i>	replaced by old active	replaced by old backup
/[x]!&	AND NOTed with <i>x</i>	replaced by old active	replaced by old backup
![x]&	inverts and ANDs with <i>x</i>	unchanged	unchanged
[*]	replaced by projection	replaced by old active	replaced by old backup

* See the "**High Speed ORing**" topic following

Selective refinement

The complex DBFIND *argument* illustrated previously may be broken up into several DBFIND calls rather than being performed in a single call.

For example, the DBFIND mode 12 *arguments* would be:

1. DBFIND *argument* = **(COMB and BIND);**
2. DBFIND *argument* = **or (HOLE and DRILL);**
3. DBFIND *argument* = **not FASTEN@;**

The result is the same as when using a single DBFIND call with the complete *argument* containing multiple values. After each DBFIND, the number of qualifying entries in the SI-chain (stored in the active SI-subset) is returned in the *status* array, and this may be reported to the user to decide at any stage whether or not to continue.

The special / operator may be used to save an intermediate result in the backup SI-subset and retrieve it later using the special \ operator.

❖ It is more efficient to specify all the arguments at one time and use a single DBFIND when possible because fewer intrinsic calls have to be made.

Positioning on a virtual SI-chain

A similar technique to that described above may be used to position at the beginning or end of or at any entry on a virtual SI-chain in the active SI-subset.

DBFIND *modes* 1nn, and 5nn may be used to position on the entry whose SI-key value matches the specified *argument*; if no matching entry exists, the internal SI-pointer is set to the location where the entry would reside. The following parameters could be used:

```
dset =      CUSTOMER-MASTER
mode =      -103
item =      ; or 0
argument =  ABC
```

❖ The “; or 0” in the item parameter signifies a “null item”. It is the null item that causes the DBFIND to act on the active SI-subset rather than the dataset.

In this example, the SI-pointer would be positioned at the SI-index for the customer ABC.

This same technique may also be used for going to the beginning of (rewinding) or end of a virtual SI-chain, using *modes* 100 and 500, respectively. The entries could then be retrieved with DBGET *modes* 15 and 16.

Determining entry count of a virtual SI-chain

In addition to being able to locate and retrieve entries on any virtual SI-chain in the active SI-subset, it is possible to determine the number of entries on any virtual SI-chain by calling DBFIND in *mode* 1 with a special *argument*, as shown:

```
base =      CUST
dset =      CUSTOMER-MASTER
mode =      1
item =      ; or 0
argument =  @@
```

This returns the entry count for the SI-chain that corresponds with the base and dataset specified in the *base* and *dset* parameters.

Finding entries using multiple SI-paths in a dataset

This same technique of using multiple DBFIND calls to refine a selection works not only on a single SI-path in a dataset but on multiple SI-paths.

For example, to find all customers that start with "GENERAL" and are located in "LOS ANGELES," two DBFIND calls would be performed and their results combined. The first DBFIND call would include these parameters:

```
item =      CUSTOMER-NAME
argument = GENERAL@;
mode =      12
```

The second DBFIND call would specify a different SI-path and the *argument* that corresponds with that SI-path, as well as a Boolean operator indicating how the results should be combined:

```
item =      CITY
argument =  and "LOS ANGELES";
mode =      12
```

The **AND** on the second DBFIND call indicates that the two sets should be ANDed. The second set can instead be logically ORed or AND NOTed, by specifying **OR** or **NOT** instead of **AND**.

For example, to find all the customers who have not placed any orders since January 1, 1991 OR have an average order amount of fifty dollars or less, two DBFIND calls are performed with the **OR** operator prefixing the *argument* on the second DBFIND:

```
item =      LAST-ORDER-DATE
argument =  <=901231;
mode =      12
```

```
item =      AVG-ORDER-AMT
argument =  or <=50;
mode =      12
```

Finding entries using multiple datasets

The same technique may be used to qualify entries across multiple datasets by using multiple DBFIND calls, each specifying a different dataset.

It is preferred that both datasets contain a common item that is used in an SI-subkey in each set. In this case, the common item forms a logical linkage between the two sets, and is referred to as the *SI-link*.



It is required that the item assigned as the SI-link be configured as an SI-subkey in the SI-path that the DBFIND is being called against; alternately, for SI-paths against a master dataset, it may be the IMAGE search master field.

If there is no common item between the datasets defined as an SI-subkey, a *projection* may be performed, as will be explained later.

Let's look at an example that locates all the customers that are slow paying for orders and currently have unpaid orders. The customers are contained in the CUSTOMER-MASTER dataset which has CUSTOMER-NUMBER as its search field and a simple SI-path called *AVG-DAYS-TO-PAY*. The orders are contained in the ORDER-DETAIL set, which has an SI-path called *ORDER-STATUS* which is comprised of the SI-key items ORDER-STATUS and CUSTOMER-NUMBER.

Both datasets have the CUSTOMER-NUMBER in common, so this defines the SI-link used to logically join the two sets. The SI-link is declared as a second value in the *item* parameter on one or both DBFIND calls, in addition to the SI-path name, in this format:

item = SI-path,SI-link

To accomplish the search, DBFIND is first called to locate all the unpaid orders with the specified SI-path and SI-link:

```
dset = ORDER-DETAIL
item = ORDER-STATUS , CUSTOMER-NUMBER
argument = UNPAID ;
mode = 12
```

Then, DBFIND is called again to locate all the customers that take an average of more than 45 days to pay. An **AND** prefixes the *argument* value to cause the virtual sets to be ANDed:

```
dset = CUSTOMER-MASTER
item = AVG-DAYS-TO-PAY , CUSTOMER-NUMBER
argument = and >=45 ;
mode = 12
```

Alternately, the virtual sets could have been ORed or AND NOTed by substituting the **OR** or **NOT** operator in place of the **AND** operator.

After both DBFIND calls are completed, DBGETs could be performed against CUSTOMER-MASTER to retrieve the qualifying entries.

In this example, there are two virtual SI-chains in the active SI-subset. For this reason, the AND is done on the values of the SI-link. If instead there were only one SI-chain in the active SI-subset, the AND would be done on the SI-extension (the search field value for masters or relative record number for details; refer to the *Internal structures* appendix for more information about the layout of the SI-subset).

The SI-link value can be assigned a length shorter than the full length of the linking field. Simply add a length parameter to the linking item (referred to as the *SI-link-length*) in the DBFIND (the length may only be specified in "words").

For example:

```
item = ORDER-STATUS, ORDER-TYPE, 3;
```

This example would use the first 3 words of ORDER-TYPE for the linking value.

Finding corresponding entries in multiple datasets

When DBFIND is called in succession against multiple datasets, one SI-chain per dataset is placed into the active SI-subset. Entries may be retrieved from any of these virtual SI-chains, independent of one another, simply by using DBGET with the appropriate dataset specified in the *dset* parameter. Since the entries on these SI-chains are logically related by the SI-link, it is often desirable to find entries on one or more of the SI-chains whose SI-link values match a specified value, thereby performing a search against a virtual SI-chain in the active SI-subset rather than against entries in a dataset.

This technique is facilitated by performing a DBFIND on each SI-chain with the exact SI-link value specified in the *argument* parameter and **;** or **0** specified in the *item* parameter. **It is the "null item" that causes the DBFIND to act on the active SI-subset rather than the dataset.** The exact SI-link value must be specified.

For example, to find all the customers whose SI-link value is equal to "ACME" call DBFIND with the following:

```
dset =      CUSTOMER-MASTER
mode =      1
item =      ; or 0
argument=   ACME
```

Subsequent DBGETs in *mode* 5 or 6 will access the sub-selected entries that have an SI-link value of "ACME," while DBGET *modes* 15 and 16 will access the entire virtual SI-chain.

Finding entries using multiple databases

Entries in multiple databases may be located in very much the same way as those in multiple datasets. Again, multiple DBFIND calls are used with an SI-link, but each has a different value for the *base* parameter. If a different item in each base is used as the SI-link, they must be configured with the same length.

SUPERDEX requires that DBFINDs against multiple databases be logically linked together. If both DBFIND calls are performed in immediate succession (with no intermediate intrinsic calls) and therefore use the same *status* array, SUPERDEX automatically links the bases together, and no specification is required by the program.

❖ **If intermediate calls are performed, the program must logically link the bases. To facilitate this, DBFIND returns a unique number in the second word of the *status* array (unused by IMAGE). The program must retrieve this number from the *status* array of the first DBFIND and specify it in the second word of the *status* array in the second DBFIND.**

Let's look at an example of two databases, one containing customers and the other sales history. We want to determine sales trends of books to schools in the CUST base by reviewing historical data in the SALES base.

The relevant sets in each base have the common item CUSTOMER-NUMBER, so this item will be used as the SI-link. If no common item exists, a *projection* may be done, as we will see shortly.

The first DBFIND call locates the customers that are schools using a keyworded SI-key. The *argument* values are partial keys being ORed together:

```
base =      CUST
dset =      CUSTOMER-MASTER
item =      CUSTOMER-ID,CUSTOMER-NUMBER
argument =  SCHOOL@ or UNIVERSITY@ or COLLEGE@;
mode =      12
```

The second DBFIND call accesses the appropriate dataset in the other database:

```
base =      SALES
dset =      PART-SUMMARY
item =      VENDOR-ID,CUSTOMER-NUMBER
argument =  and (MCMIL@ or MCGRAW@);
mode =      12
```

❖ Both calls specify the SI-link as the second value in the *item* parameter. Also, the use of **AND** at the beginning of the *argument* in the second DBFIND call tells SUPERDEX to AND its results with the previous DBFIND call.

As always, the **OR** or **NOT** operator could have been used instead of the **AND** operator to perform an OR or AND NOT retrieval between databases.

Finding entries in multiple sets and bases using projection

In the last two examples, a common item exists between the two datasets and databases being searched, and was defined as the SI-link. For situations in which there is no common item but a logical relationship exists, *projection* may be used.

A projection is an operation that permits two datasets that do not contain a common item to be linked together, providing each has an item in common with a third dataset. For this discussion, we'll refer to the first dataset as set *A*, the second dataset as set *B*, and the third (linking) dataset as set *C*. The projection reassigns the SI-link from the item set *C* has in common with set *A* to its item in common with set *B*, thereby forming a logical relationship between set *A* and set *B*, even though they do not contain a common item.

A projection is invoked by a separate DBFIND against set *C* which is called between the DBFINDs against set *A* and set *B*. Internally, a projection takes the SI-link values returned by the DBFIND against set *A* (which are stored in the active SI-subset), looks up the corresponding entries in set *C*, and replaces them with the SI-link values that will be used for set *B*. The SI-link for set *A* and set *B* are both defined by item name or number in the *item* parameter, as shown:

```
item =      SI-path,new SI-link
```


where *SI-path* is a concatenated SI-path that is comprised of the *old SI-link* as SI-subkey-1 and the *new SI-link* as any other SI-subkey.

Let's look at an example that locates all quotations for earthquake coverage given to policyholders in Los Angeles in November and December of 1987. This requires four DBFIND calls against three datasets, with the second DBFIND call performing the projection.

The first DBFIND locates all policyholders in Los Angeles, with SI-link specified in the *item* parameter:

```
dset =      INSURED-MASTER
item =      CITY , POLICY-NUMBER
argument =  "LOS ANGELES" ;
mode =      12
```

The next DBFIND performs the projection, as designated in the *argument* by the special * (asterisk) operator. The *item* specifies the SI-link used in the previous DBFIND, as well as the SI-link that will be used in the next DBFIND:

```
dset =      QUOTE-MASTER
item =      POLICY-NUMBER , QUOTE-NUMBER
argument =  [ * ]
mode =      1
```

The third DBFIND accesses the same dataset in which the projection was performed and locates all the entries in the specified date range. The new SI-link is specified in the *item* parameter:

```
dset =      QUOTE-MASTER
item =      QUOTE-DATE , QUOTE-NUMBER
argument =  and >=871101<=871231 ;
mode =      12
```

The final DBFIND locates all the entries in another dataset that are of the requested coverage type:

```
dset =      QUOTE-DETAIL
item =      COVERAGE-TYPE , QUOTE-NUMBER
argument =  and EQ ;
mode =      12
```

Notice that the last two DBFIND calls used **ANDs** in the *argument* to logically AND the results.

❖ Projection uses all qualifying SI-link values contained in the active SI-subset in locating entries. If there is more than one SI-chain in the active SI-subset, entries may qualify more than once. Therefore, a projection should only be performed when there is only one SI-chain in the active SI-subset.

Circumstances in which the SI-link must be specified

As described, the SI-link is a common item that is configured in an SI-subkey in each set and which is used to form a logical linkage between the two sets. It is required that the item assigned as the SI-

link be configured as an SI-subkey in a concatenated SI-key; alternately, for SI-paths against a master dataset, it may be the IMAGE search field.

The SI-link need not be specified in all cases involving relational access against multiple datasets and/or databases, and may be omitted on some DBFIND calls. The following rules govern the specification of the SI-link:

1. For relational access between two different SI-paths within the same dataset, the SI-link is not required. The Boolean operations are based only on the SI-extension (the search field value for a master or relative record number for a detail).
2. For relational access between two datasets in the same database, the SI-link must be specified for the first DBFIND call but not the second, although there is no harm in specifying it for both DBFIND calls. In this case, the value of the SI-link is used for comparison instead of the SI-extension. The SI-link need not be specified for a manual master for which the IMAGE search field is the default SI-link.
3. For relational access between databases, the SI-link is always used for comparison and must always be specified for every DBFIND call. Additionally, if intermediate calls between databases are performed, the bases must also be logically related via word 2 of the *status* array, as described previously.

Qualifying entries in the active SI-Subset

While SI-Subsets were designed for relational retrievals, they can be very useful for ultra fast access since they can be considered as in-memory datasets.

An in-memory dataset is created by a DBFIND call, where the SI-Link is specified and the argument is a relational argument. This SI-Subset acts like a dataset with SI-Link as a SUPERDEX key. You can access this set by DBFIND utilizing most of the SUPERDEX features.

For example, first create an SI-Subset for customer "ACME".

```
dataset =      CUSTOMER-MASTER
mode =         12
item =         CUSTOMER-NAME,ORDER-NUMBER
argument =     ACME;
```

Now do a selection on this SI-Subset with order-number 70123:

```
dataset =      CUSTOMER-MASTER
mode =         1
item =         ; or 0;
argument =     70123
```

Instead of using mode 1, the special modes 1nn or 5nn can be used to retrieve partial keys in sorted order. **Partial key retrieval using the @ operator and the ? or # match characters are NOT available.**

❖ **In-memory datasets can be especially useful for applications where you expect many DBFINDs to fail, or when you need the same information frequently, since it would be returned without further disc access.**

Preparing the argument

The many *argument* operators available with SUPERDEX allow very advanced and powerful retrievals using the simple DBFIND *mode* 1 or using the relational DBFIND *mode* 12. *Arguments* can contain one or more of the following in various combinations:

- the @, <<, >>, ?, and # conditional operators
- the <=, >=, <> and == retrieval operators
- the AND, OR, and NOT Boolean operators
- the +, , and - Boolean operators
- the &, |, and !& Boolean operators
- the *, /, \, \$, and ^ special operators

Allowing these generalized *arguments* for DBFIND *modes* 1, 12, 21, and 23 permits standard lookup routines to perform many types of retrievals, with the *argument* determining the type and scope of access. Complex *arguments* may be prepared for DBFIND calls by several methods.

The simplest method, which requires little or no reprogramming, is to have users specify the entire *argument* themselves, including multiple values and various operators, and have the program pass them literally to DBFIND.

Another method is to assign function keys to facilitate various retrieval capabilities. The user could enter a string and hit a function key, and the program could read the function key label and concatenate the appropriate operator to the specified value to form the *argument*.

Various other methods are available for programmatically constructing the DBFIND *argument*, such as prompting the user with various selection boxes to check off or enter values into.

High Speed OR'ing

If a DBFIND contains an argument with two or more values to be OR'ed together, and then AND'ed to the previous DBFIND, an overflow can occur during ORing even when it is possible that the net result of qualified records would fit in the table. Two DBFIND operators have been added to make this process possible; [] and \$. The [] will automatically copy the active SI-subset to the backup SI-subset and clear the active SI-subset. The \$ will then AND the search value to the backup SI-subset, joining the qualified entries into the active SI-subset.

The net result of this process is that the records qualified by the additional DBFIND will not automatically be written to the active SI-subset, but will first be qualified against the backup SI-subset. This means that the active SI-subset will never contain more records than the backup SI-subset, eliminating most table overflows.

To illustrate the process:

1. DBFIND, *mode* 12, on path NAME with argument **ABC COMPANY@;** places the records that qualified into the active SI-subset.
2. DBFIND, *mode* 1, with argument [] will replace the backup SI-subset with the active and create an empty active SI-subset.

3. DBFIND, *mode* 1, on path DATE with argument [91@]\${92@}\$ will process each argument separately. First all the records in the backup SI-subset will be AND'ed with the DATE of 91@ into the active SI-subset, then the records in the backup SI-subset will be AND'ed with the DATE of 92@ joined into the existing active SI-subset.

Steps 2 and 3 could be combined into a single DBFIND on path DATE with an argument of [][91@]\${92@}\$.

Before these new operators, steps 2 and 3 would have been in a single DBFIND on path DATE with argument [91@][92@] |&.

Effect of DBFIND on the SI-pointer and current path

Like IMAGE, SUPERDEX returns a condition word of zero for successful DBFIND calls and non-zero if an error or exception is detected.

In IMAGE, calling DBFIND against a master set returns condition word -21 ("**SPECIFIED INTRINSIC IS NOT ALLOWED ON MASTER SET**"); in SUPERDEX, no error is returned because it is a valid operation.

❖ Since the condition word -21 is not returned on the master dataset for a DBFIND, some generic access programs may need modification. If a DBFIND is used to determine whether a dataset is a master or detail, this will not work. If a master dataset has at least one SI-path, the DBFIND will not return the -21, but can return the condition word -52 (see following).

If the specified *item* is neither a valid SI-path nor IMAGE path, condition word -52 ("**ITEM SPECIFIED IS NOT AN ACCESSIBLE SEARCH ITEM IN THE SPECIFIED SET**") is returned.

If DBFIND does not find an entry that matches the specified *argument*, condition word 17 ("**NO ENTRY FOUND**") is returned. If the SI-path name is the same as the IMAGE search field name and the SUPERDEX DBFIND against the SI-path fails, an IMAGE DBFIND is automatically performed against the search field. If called in *mode* 1, 12, 21, or 23, the SI-pointer is not set and the current path is reset to the dataset's current IMAGE path. Therefore, if the condition word is ignored and subsequent DBGETs are called, they will operate on an IMAGE path (the current IMAGE path for a detail set or the synonym chain for a master set) rather than the SI-path.

If DBFIND is called with *mode* 1nn, 2nn, 3nn, 4nn, or 5nn, the SI-pointer is set immediately before or after the nearest qualifying entry and the current path is set to the appropriate SI-path and does not change. For example, DBFIND *mode* 102 with an *argument* of BRAC may not locate a matching entry but will set the SI-pointer before "BRADMARK," the nearest qualifying entry. Then, DBGET *mode* 15 or 16 may be used to retrieve the entries in ascending or descending order.

The following table summarizes the effects of DBFIND on the SI-pointer and current SI-path:

<i>mode</i>	<i>condition word = 0</i>	<i>condition word <> 0</i>
1	before entry	current path not set ***
12	before entry	current path not set ***
21	before entry	current path not set **
23	before entry	current path not set **
100*	before first (alphabetical) entry	current path not set
1nn	before entry	if <i>cw</i> = 17, before next entry
200*	before first (alphabetical) entry	current path not set
2nn	before entry	if <i>cw</i> = 17, before next entry
300*	before first (alphabetical) entry	current path not set
3nn	before entry	if <i>cw</i> = 17, before next entry
400	after last (alphabetical) entry	current path not set
4nn	after entry	if <i>cw</i> = 17, after next entry
500	after last (alphabetical) entry	current path not set
5nn	after entry	if <i>cw</i> = 17, after next entry

- * may be followed by either DBGET *mode* 5 or 15 or *mode* 6 or 16 and will start at either the beginning or end of the dataset, respectively, as compatible with IMAGE
- ** if using relational access, the current path is retained
- *** if using relational access, the current path is retained; otherwise, defaults to current IMAGE path

CHAPTER 3:

Retrieving entries with DBGET

Entries that are located with DBFIND may be returned in ascending sorted sequential order with *modes* 5 and 15 and descending order with *modes* 6 and 16. There are additional DBGET modes described in the *TurboIMAGE Third-Party Indexing External Specifications*. This document, written by Hewlett-Packard, is contained in its entirety in *Appendix A*. All of the additional modes are supported within SUPERDEX, but are not required. Therefore this chapter only describes standard SUPERDEX modes.

Like IMAGE, *modes* 5 and 6 return condition words 14 and 15 ("**BEGINNING OF CHAIN**" and "**END OF CHAIN**") when all qualifying entries on the SI-chain have been returned. *Modes* 15 and 16 continue to return entries in sorted sequential order that are not part of the SI-chain, like greater-than-or-equal-to and less-than-or-equal-to retrievals.

DBGETs with Un-initialized SI-chain

If the SI-chain has not been established or the SI-pointer is outside of the current SI-chain, this could be the result of:

- DBFIND with *mode* from 100 to 599
- DBFIND that does not find a match (returns condition word 17)
- DBGET *mode* 4

In these cases, DBGET *modes* 15 and 16 should be used instead of *modes* 5 and 6; otherwise, the results are unpredictable.

Repositioning on an SI-chain

If reading an SI-chain (not an SI-subset) along an SI-path that has a *unique* relationship (one SI-index per data record, unlike with a keyworded SI-path), DBGET *mode* 4 can be used to reposition on the SI-chain. An application for this is, for example, implementing a "previous page" function when displaying entries on a terminal. This can be programmed by keeping an internal list of relative record numbers of the first entry on each page and then returning to any page by calling DBGET *mode* 4 followed by DBGET *mode* 5s in a loop.

To reposition on an SI-chain after switching to a different SI-path or IMAGE path in the same dataset, save the relative record number before switching paths, then to return, call DBFIND *mode* 23 with the original search argument followed by DBGET *mode* 4 using the saved record number. From here, DBGET *modes* 5, 6, 15, and 16 may be performed normally.

Additionally, DBGET *mode* 11 will position the pointer to the beginning of the qualified list, and *mode* 12 will move the pointer forward "*n*" entries, where "*n*" is a 32-bit integer sent in the *argument*. DBGET *mode* 13 can be used to move the pointer backwards "*n*" entries.

Reading SI-indices only

It is desirable for efficiency to restrict the DBGETs to reading only the SI-indices rather than actually retrieving the entries from the datasets whenever possible.

When performing indexed access, the full SI-index including the SI-extension (search field value for a master dataset or relative record number for a detail) is returned. For relational access, the SI-link (if specified) and SI-extension are returned.

Several operations that may be accomplished by reading SI-indices:

- validating the format of any value in an SI-index, which always includes the search field value for SI-paths related to master datasets.
- intermediate storage of the SI-extension to facilitate resetting the SI-pointer to its former position, such as after changing SI-paths
- testing for the presence or absence of a value in multiple SI-paths by reading them in parallel

To read SI-indices only, use a list of **!** with DBGET *modes* 5, 6, 15, or 16.

To retrieve the entry associated with any SI-index, use DBGET *mode* 1 (reread current entry).

❖ **The **!** list is never transferred to IMAGE; therefore, the list in use before the **!** was declared will still be active.**

Reading multiple SI-indices with a single DBGET

When using the **!** list to read SI-indices only, it is possible to read multiple SI-indices with a single DBGET, equivalent to calling DBGET multiple times in a loop.

This is facilitated by an optional *SI-counter* parameter, which specifies the number of SI-indices to return. If not specified, the default SI-counter value is 1. The SI-counter is a numeric literal immediately following the **!**, as shown:

```
list =      !SI-counter
list =      !50
```

If a beginning-of-chain or end-of-chain condition is detected during the DBGET call, the returned indices are less than the requested number of indices. The number of indices returned is contained in word 5-6 of the status array. The SI-Pointer is positioned at the first/last index.

❖ **If using an SI-counter, be sure to use a *buffer* large enough to accommodate all the SI-indices that will be returned.**

❖ The SI-counter parameter cannot be used with any *list* construct other than !.

Effect of DBGET on the SI-pointer and current path

Like IMAGE, SUPERDEX returns a condition word of zero in the *status* array for successful DBGET calls and non-zero if an error is detected.

In addition, DBGET *modes* 5 and 6 performed on a master set read down and up the SI-chain, whereas in IMAGE, they read down and up the current synonym chain.

If the *item* specified is neither a valid SI-path nor IMAGE path, condition word -52 ("ITEM SPECIFIED IS NOT AN ACCESSIBLE SEARCH ITEM IN THE SPECIFIED SET") is returned.

The following table summarizes the effects of DBGET on the SI-pointer and current SI-path:

<i>mode/relationship</i>	<i>condition word = 0</i>	<i>condition word <> 0</i>
5 or 15*	before next entry	if <i>cw</i> = 15, no change
6 or 16*	after previous entry	if <i>cw</i> = 14, no change
other unique	on entry	no change
other non-unique	no change	no change

* if switching from *mode* 5 or 15 to *mode* 6 or 16, the same entry is not returned twice.

Serially Reading All Entries

Like IMAGE, SUPERDEX will allow a serial read through a dataset using DBGET *mode* 2. This will return the records in the same order as IMAGE.

If you wish to sequentially access all the records in sorted order by any path, initialize the path using DBFIND *mode* 100, and retrieve them with DBGET *mode* 15. This will return the records in sorted order, based on the SI-path defined, and return a condition word of 11 (End-of-file), as DBGET *mode* 2 does.

For master datasets, using DBFIND *mode* 100 and DBGET *mode* 15 can be faster and more efficient than DBGET *mode* 2. SUPERDEX does not need to read each block in the dataset, while IMAGE needs to read each block to see if any records exist in the block.

❖ Remember that all entries are returned based on the indexes. Therefore, many-to-one indexes, such as grouped or keyworded, will return each user record every time an index entry for that record is found.

CHAPTER 4:**Additional programming considerations****Summary of effects of SI-intrinsics on the SI-pointer and current SI-path**

The table below indicates the effects of various intrinsics on the SI-pointer and current SI-path.

In performing a DBPUT, DBPUTIX, DBDELETE, DBDELIX, or DBUPDATE against an SI-path for which more than one SI-index may point to the same record (a *n*-to-one relationship referred to below as a *non-unique* relationship as in keywording, grouping, and SIUSER), the position of the SI-pointer does not change--regardless of whether or not a unique relationship exists for a given entry.

<i>intrinsic/mode</i>	<i>condition word = 0</i>	<i>condition word <> 0</i>
DBFIND mode 1	before entry	current path not set ***
DBFIND mode 10	before entry	current path not set ***
DBFIND mode 100*	before first entry	current path not set
DBFIND mode 1nn	before entry	if <i>cw</i> = 17, before next entry
DBFIND mode 500	after last entry	current path not set
DBFIND mode 5nn	after entry	if <i>cw</i> = 17, after next entry
DBGET modes 5/15**	before next entry	if <i>cw</i> = 15, no change
DBGET modes 6/16**	after previous entry	if <i>cw</i> = 14, no change
DBGET other modes unique	on entry	no change
DBGET other modes non-unique	no change	no change
DBPUT	no change	no change
DBPUTIX	no change	no change
DBUPDATE	no change	no change
DBDELETE	no change	no change
DBDELIX	no change	no change

* may be followed by either DBGET *mode* 5 or 15 or *mode* 6 or 16 and will start at either the beginning or end of the dataset, respectively, as compatible with IMAGE

** if switching from *mode* 5 or 15 to *mode* 6 or 16, the same entry is not returned twice

*** if using relational access, the current path is retained

Testing for the existence of SUPERDEX;

The DBINFO *mode* 801 is provided to test for the presence of SUPERDEX, and returns information about the SUPERDEX version configured for a specified database. It is designed for programs that are run against the same database on various systems, of which some do not contain SI-paths. The name of the indexing tool, version, etc... are returned via DBINFO mode 801 (see DBINFO in [Section 5: Intrinsics](#)).

PowerHouse

- ◆ SUPERDEX has interfaces to Cognos' PowerHouse modules QUICK, QUIZ, and QTP available, described in the Fourth-Generation Language manual. Contact your Bradmark sales representative or distributor for information and a demonstration.
-

SUPERDEX also supports a special PowerHouse Date index. For information on this index option, refer to *Section 3*.

TRANSACT

Programs written in TRANSACT are converted quite easily to use SUPERDEX. Basically, the SI-path name is placed into the *key* register, the search argument into the *argument* register, and **FIND(CHAIN)**, **FIND(RCHAIN)**, or **PATH** is called against a master or detail dataset. Please see the *TRANSACT Interface* supplement for more information and documentation.

PROTOS

PROTOS fully supports the SUPERDEX product through their own interface. For more information, contact Bradmark Technologies, or PROTOS.

VISIMAGE

VISIMAGE fully supports the SUPERDEX product through their own interface. For more information, contact Bradmark Technologies.

CHAPTER 5:**Native Language Support**

Adding, updating, and indexing entries

When Native Language Support (NLS) is activated for a database, SI-indices are generated by SIMAINT, DBPUT, DBPUTIX, DBUPDATE and SI-USER according to HP's documented NLS collating sequences. This may result in some confusion because indexing may be done differently with and without NLS.

The following table shows how SI-indices for a concatenated SI-key would be sorted with and without NLS:

<i>with NLS</i>	<i>without NLS</i>	<i>Keywords without NLS</i>
PAPER 100	PAPER 100	PAPER 100
PAPER 400	PAPER 400	PAPER 200
Paper 300	PAPERBAG 150	PAPER 300
paper 200	Paper 300	PAPER 400
PAPERBAG 150	paper 200	PAPERBAG 150

Qualifying entries with DBFIND

To qualify the entries with the SI-keys shown above, the *arguments* PAPER@, Paper@, and paper@ are treated equivalently and all of the above entries are returned in the order shown in the left column.

❖ Since the different representations of the same word are stored as distinct indices, there are circumstances where the same entry may qualify more than once.

SECTION 5:

Intrinsics

Overview

This section describes the various enhancements provided with SUPERDEX through TurboIMAGE as well as the new DBERASE, DBPUTIX, DBDELIX intrinsics and SIUSER procedure.

It does not describe **ALL** of the new modes and possibilities with SUPERDEX and TurboIMAGE. Refer to Appendix A for the complete description of the TurboIMAGE Third-Party Indexing Interface Specifications.

Chapter 1 Enhancements

Function Documents the Enhancements to TurboIMAGE.

Documents each intrinsic, including a discussion of each, their syntax, and parameters. The intrinsics are listed in alphabetical order by name.

CHAPTER 1:

Enhancements

The enhancements to the standard TurboIMAGE intrinsics are summarized alphabetically by intrinsic in the following table. Except for the new SUPERDEX intrinsics, the TurboIMAGE intrinsics are functionally and syntactically identical to TurboIMAGE without SUPERDEX, so the changes are transparent; therefore, only variations are documented here.

Summary of intrinsic enhancements and additions

DBCONTROL	new 800 modes
DBDELIX	new intrinsic; explicitly deletes SI-index entry
DBERASE	new intrinsic; erases dataset and associated SI-indices in a fast mode
DBFIND	<ul style="list-style-type: none">- works the same on master and detail dataset- new <i>mode</i> 21 works same as <i>mode</i> 1 but does not return qualifying entry count- new <i>mode</i> 12 enforces that the argument will be treated relationally- new <i>modes</i> 100-599 position SI-pointer- <i>argument</i> may contain multiple values and operators- multiple calls perform dynamic queries on multiple fields, sets, and bases- can qualify a master entry based on its related detail entries in a super-group- the results of the current and previous DBFIND are maintained
DBGET	<ul style="list-style-type: none">- works the same on master and detail datasets- <i>modes</i> 5 and 6 retrieve entries in ascending and descending sorted order- new <i>modes</i> 15 and 16 return all entries in the dataset alphabetically- new <i>! list</i> reads SI-indices only; optional <i>SI-counter</i> returns multiple SI-indices
DBINFO	new modes return information about SUPERDEX configuration
DBOPEN	new 100 modes
DBPUTIX	new intrinsic; explicitly adds SI-index entry
SITRANSLATE	new procedure to convert an <i>argument</i> from Infix Notation to Reverse Polish Notation
SIUSER	new user-written procedure; permits customer-defined SI-indices

DBCCONTROL intrinsic

❖ Remember that only SUPERDEX requirements are documented. For complete information on the Third-Party Indexing Interface External Specifications, see *Appendix A*.

DBCCONTROL is used to turn certain capabilities on and off.

Syntax **DBCCONTROL** (*base,dset,mode,status*)

Parameters

Base The *base-ID* (same as IMAGE).

Dset For the 800 DBCCONTROL modes, this parameter is ignored.

Mode **800** If DBFIND *mode* 12 qualifies 0 records, the active SI-subset will be replaced with the backup SI-subset automatically.
801 If DBFIND *mode* 12 qualifies 0 records, the active SI-subset will be left empty (**DEFAULT** mode).
802 Exposes the SI-dataset(s) for DBINFOs.
803 Hides the SI-dataset(s) for DBINFOs (**DEFAULT** mode).

Status Same as current IMAGE.

DBDELIX intrinsic

DBDELIX is a SUPERDEX intrinsic used to explicitly delete SI-indices from B-trees (its counterpart is the SUPERDEX DBPUTIX intrinsic, which explicitly adds SI-indices).

The DBDELIX intrinsic accesses only the appropriate SI-dataset and is used to maintain independent indices. It also provides a method for deleting custom SI-indices in addition to those removed automatically by DBDELETE.

Refer to the *Adding, updating, and deleting entries* chapter of the *Programming* section for further discussion.

Syntax **DBDELIX** (*base,dset,mode,status,item,buffer*)

The DBDELIX intrinsic is syntactically similar to DBDELETE except that the *list* parameter is replaced by an *item* parameter and the *buffer* parameter contains the full SI-index, including the SI-extension.

Parameters

Base	The <i>base-ID</i> (same as IMAGE).
Dset	Name or number of the dataset in which the corresponding data entry exists. If accessing an independent SI-path, this parameter should be left blank or set to 200 .
Mode	An integer with the value 1 .
Status	Only the condition word is set.
Item	The name of the SI-path from whose B-tree to delete the specified SI-index.
Buffer	The full SI-index value including the extension. The extension may be: <ul style="list-style-type: none">■ the search field value for a master dataset■ the relative record number for a detail dataset■ a suitable user-defined value for independent SI-paths

Error handling

Since SUPERDEX uses standard IMAGE messages to report all errors and exceptional conditions, the same messages that are used by DBDELETE are displayed for DBDELIX.

DBERASE intrinsic

DBERASE is a SUPERDEX intrinsic that erases the contents of a dataset and also removes all corresponding SI-indices for all associated SI-paths. It is considerably faster than DBDELETEing all entries.

All entries in the dataset are read serially and erased, then the corresponding SI-indices are erased. As with DBDELETE, a master dataset cannot be erased if any of its entries have related detail entries--if they do, the erase will not be performed at all.

Syntax **DBERASE** (*base,dset,mode,status*)

Parameters

Base The *base-ID* (returned by DBOPEN).

Dset The name or number of the master or detail dataset to erase.

Mode An integer with the value **1**.

Status Standard IMAGE *status* array. Only the condition word is set (to zero if successful or to one of the IMAGE condition words returned by DBDELETE if unsuccessful).

◆ Remember that a dataset lock must be issued prior to calling DBERASE, and that DBERASE cannot be used if the dataset contains a Supergrouped index.

Improved speed in exclusive access mode

Although erasing a dataset with DBERASE is much faster than using DBDELETES, its speed can be increased further if run in exclusive access mode by logging on as the database creator and executing the following DBUTIL commands:

```
:RUN DBUTIL.PUB.SYS

>>DISABLE base FOR ILR

  ILR is disabled.
>>DISABLE base FOR LOGGING

  Logging is disabled.
>>ENABLE base FOR AUTODEFER

  Autodefer is enabled.
>>EXIT

END OF PROGRAM
```

Don't forget to reset these run-time options back to their original status using the ENABLE and DISABLE commands after the DBERASE has completed.

Recovery after abnormal abort

If the DBERASE fails due to a program abort, system failure, or other interruption, erase the dataset using a utility (such as DBGENERAL) or delete any remaining entries using QUERY.PUB.SYS and use the SIMAINT utility to reorganize all SI-paths related to the dataset. If SIMAINT is unable to successfully recover the database, use SIMAINT with the DBLOAD entry-point.

DBFIND intrinsic

❖ Remember that only SUPERDEX requirements are documented. For complete information on the Third-Party Indexing Interface External Specifications, see *Appendix A*.

DBFIND accesses an SI-path and sets the SI-pointer within that SI-path's B-tree for subsequent DBGETs. In addition to standard TurboIMAGE, DBFIND works on both master and detail sets.

In using DBFIND against SI-paths, the *argument* may contain partial keys as well as generic values, relational operations, ranges, and multiple values which are logically combined via Boolean operators. For keyworded SI-paths, the *argument* may contain a keyword; for concatenated SI-keys, it may contain a concatenated value; both may include partial and generic keys, etc.

If qualifying entries using a super-grouped SI-path, DBFIND must be called against the master set, although entries in all datasets in the super-group will be used to qualify the master entries.

SUPERDEX allows multiple DBFIND calls in succession to qualify entries across multiple SI-paths, datasets, and even multiple databases. A similar technique may be used to refine a selection, whereby further qualification may be performed against entries already found.

Refer to the *Qualifying entries with DBFIND* chapter of the *Programming* section for further discussion and examples.

Syntax DBFIND (*base, dset, mode, status, item, argument*)

Parameters

Base Same as IMAGE

Dset In addition to IMAGE, master sets as well as detail sets may be specified by name or number. If the SI-path is super-grouped, the master set must be specified.

If accessing an independent SI-path, this parameter should be left blank or set to **200**.

Mode While IMAGE allows only a single *mode* (*mode 1*) for DBFIND, SUPERDEX extends the capabilities of *mode 1* while maintaining compatibility, and provides several additional *modes* for use on SI-paths.

Mode 1 Fully compatible with IMAGE *mode 1*. Also allows multiple values and various operators to be included in the *argument* (explained later). If no entry is found that matches the *argument*, condition word 17 ("NO ENTRY FOUND") is returned.

❖ **DBFIND mode 1 returns the number of qualifying entries in words 5-6 of the status array.**

Mode 10 Forces a standard TurboIMAGE DBFIND on a dataset and search item that have a matching SUPERDEX SI-path. This is used to by-pass the SUPERDEX DBFIND in all cases and should be used when the search item is passed as a numeric value and the TurboIMAGE DBFIND should be used.

Mode 11 This mode provides range retrieval on binary data (I, J, K, P, R, E, Z).

❖ **DBFIND mode 11 returns the number of qualifying entries in words 5-6 of the status array.**

Mode 12 The same as *mode 1*, but forces a relational access on the DBFIND. This is used when the DBFIND should execute a relational access without the users needing to know.

❖ **DBFIND mode 12 returns the number of qualifying entries in words 5-6 of the status array.**

Mode 13 Used to restore the most recently qualified list from a previous DBFIND *mode 12*.

Mode 21 Same as *mode 1*, but does not return the number of qualifying entries in the *status* array and is therefore more efficient.

In *mode 1*, not only is the SI-pointer set to the first qualifying SI-index entry, but the B-tree is traversed to locate all qualifying SI-indices. It is more efficient to only set the SI-pointer, so *mode 21* should be used in place of *mode 1* whenever the number of qualifying entries is not required.

In *mode 21*, the chain entry count in the *status* array is always set to 1 (if the chain is not empty) for compatibility with programs that call DBFIND and test to make sure that the chain count is not zero.

Mode 22 Same as *mode 11*, but does not return the number of qualifying entries in the *status* array and is therefore more efficient.

In *mode 11*, not only is the SI-pointer set to the first qualifying SI-index entry, but the B-tree is traversed to locate all qualifying SI-indices. It is more efficient to only set the SI-pointer, so *mode 22* should be used in place of *mode 11* whenever the number of qualifying entries is not required.

In *mode 22*, the chain entry count in the *status* array is always set to 1 (if the chain is not empty) for compatibility with programs that call DBFIND and test to make sure that the chain count is not zero.

- Mode 23** Same as *mode 12*, but does not return the number of qualifying entries in the *status* array and is therefore more efficient.
- Mode 100** Positions the SI-pointer before the first SI-index in the B-tree (i.e. before the lowest alphabetical entry in the set in ascending order). In this *mode*, the *argument* is ignored and may be left blank.
- Mode 1nn/-1nn** Reads *nn* words of the *argument* and sets the SI-pointer before the first qualifying SI-index entry. It also creates a qualified list of all entries equal-to (=) the *argument* value. If no matching entry exists, condition word 17 ("NO ENTRY FOUND") is returned but the SI-pointer is set immediately before the nearest-matching SI-index. If prefixed with a minus sign (-), reads *nn* bytes instead of words.
- Mode 200** Positions the SI-pointer before the first SI-index in the B-tree (i.e. before the lowest alphabetical entry in the set in ascending order). In this *mode*, the *argument* is ignored and may be left blank.
- Mode 2nn/-2nn** Reads *nn* words of the *argument* and sets the SI-pointer before the first qualifying SI-index entry. It also creates a qualified list of all entries greater-than (>) the *argument* value. If no matching entry exists, condition word 17 ("NO ENTRY FOUND") is returned but the SI-pointer is set immediately before the nearest-matching SI-index. If prefixed with a minus sign (-), reads *nn* bytes instead of words.
- Mode 300** Positions the SI-pointer before the first SI-index in the B-tree (i.e. before the lowest alphabetical entry in the set in ascending order). In this *mode*, the *argument* is ignored and may be left blank.
- Mode 3nn/-3nn** Reads *nn* words of the *argument* and sets the SI-pointer before the first qualifying SI-index entry. It also creates a qualified list of all entries greater-than-or-equal-to (>=) the *argument* value. If no matching entry exists, condition word 17 ("NO ENTRY FOUND") is returned but the SI-pointer is set immediately before the nearest-matching SI-index. If prefixed with a minus sign (-), reads *nn* bytes instead of words.
- Mode 400** Positions the SI-pointer after the last entry, in ascending order (i.e. after the highest alphabetical entry in the set). In this *mode*, the *argument* is ignored and may be left blank.
- Mode 4nn/-4nn** Reads *nn* words of the *argument* and sets the SI-pointer after the last qualifying SI-index entry. It also creates a list of qualified entries that are less-than (<) the *argument* value. If no matching entry exists, condition word 17 ("NO ENTRY FOUND") is returned but the SI-pointer is set immediately after the nearest-matching SI-index. If prefixed with a minus sign (-), reads *nn* bytes instead of words.
- Mode 500** Positions the SI-pointer after the last entry, in ascending order (i.e. after the highest alphabetical entry in the set). In this *mode*, the *argument* is ignored and may be left blank.

Mode 5nn/-5nn Reads *nn* words of the *argument* and sets the SI-pointer after the last qualifying SI-index entry. It also creates a list of qualified entries that are less-than-or-equal-to (\leq) the *argument* value. If no matching entry exists, condition word 17 ("NO ENTRY FOUND") is returned but the SI-pointer is set immediately after the nearest-matching SI-index. If prefixed with a minus sign (-), reads *nn* bytes instead of words.

Status Same as IMAGE, although the chain count (words 5-6) is set by mode 1, 11, and 12 only; in mode 21, 22, or 23 the chain count is set to a constant value of 1 (if the chain is not empty). In relational access mode, the chain count reflects the total number of SI-indices in the active SI-subset.

The first-on-chain and last-on-chain pointers (words 7-10) are set only for IMAGE paths and not SI-paths, unless the SISETLINK JCW has been set to 1. If the SISETLINK JCW has been set, the first-on-chain and last-on-chain values will contain the correct values for the given SI-path.

Additionally, a unique internal number utilized in performing logically-related DBFINDs against multiple databases is either returned or specified in word 2, which is unused by IMAGE.

SUPERDEX returns a condition word of 0 in the first word on a successful call and a non-zero condition word on an unsuccessful call, like IMAGE; however, a SUPERDEX DBFIND with a *mode* of 1nn thru 5nn may return a condition word 17 error ("NO ENTRY FOUND") while still setting the SI-pointer before or after the nearest qualifying entry, respectively (this feature is used for approximate match retrieval too).

Item Specifies either an IMAGE path or an SI-path. If an IMAGE path, the name or number of the IMAGE search field is specified as usual. If an SI-path, the name or number of the SI-path or the item number of the first SI-subkey in the SI-key is specified.

❖ If there is more than one SI-path that starts with the same item number, or if the SI-path is an independent SI-path, the SI-path name or number must be specified.

❖ If the SI-path number is to be used, it is recommended that the numbers NOT be hard-coded in the program. A DBINFO mode 833 should be called against the path to retrieve the SI-path number. SI-path numbers are assigned dynamically, based on the configuration of the database. Therefore, if SI-paths are added or deleted the SI-path number for any SI-path may change.

If both an IMAGE path and an SI-path with the same name exist in the same dataset and DBFIND is called in *mode* 1, SUPERDEX will use the SI-path instead of the IMAGE path. If the search of the SI-path is unsuccessful, the IMAGE path is used. If the IMAGE-path search is unsuccessful, the condition word is set to 17.

In performing successive DBFINDs against multiple datasets, a common item used to logically link the datasets together (called the *SI-link*) may additionally be specified. It is required that the item assigned as the SI-link be configured as an SI-subkey; alternately, for SI-paths against a master dataset, it may be the IMAGE search field.

The SI-link is separated from the SI-path name by a comma, with the combined value terminated by a **SPACE** or **;** as shown:

SI-path,SI-link;

If the SI-path is passed as an item number rather than a name, the item number of the SI-link should be specified in the second word of the *item* array.

If performing a projection, which is used to logically link two datasets that do not contain a common item by reassigning the SI-link, the *item* parameter takes the form:

old SI-link,new SI-link;

To locate entries that have been found by previous DBFINDs in the active SI-subset rather than in the dataset, specify a null *item* of **0** or **;**.

Argument

In IMAGE, the DBFIND *argument* must specify an exact search field value. In SUPERDEX, the *argument* for DBFIND modes 1, 12, and 21 may contain:

- an exact SI-key value (or concatenated value)
- a single **@**
- a partial SI-key value with one or two **@** surrounded with **<<** **>>**
- a generic SI-key value containing one or more embedded **?**s (the alphanumeric matchcode) or **#**s (the numeric matchcode)
- a partial SI-key value preceded by either the **>=**, **<=**, or **<>** operators
- a range of two or more values with embedded operators (e.g. **>=A@<=B@**)
- an ASCII value prefixed by **==**, which causes the value to be converted to binary for comparison
- multiple SI-key values, with the argument beginning with the tilde (**~**) and ending with the semi-colon (**;**), and including one or more of the Boolean operators **AND**, **OR**, or **NOT** (specified in SQL Notation)
- multiple SI-key values, with the argument beginning with the tilde (**~**) and ending with the semi-colon (**;**), and including one or more of the Boolean operators **+**, **,**, or **-** (specified in Infix Notation)
- the special operators, such as **/**, ****, and **\$**, used for manipulating the active SI-subset and backup SI-subset
- the special operator **^**, used for swapping the active and backup SI-subsets
- the special operator *****, used for projection

- the special operator @@, which rewinds the virtual SI-chain, and in *mode 1* returns its entry count
- many combinations of these constructs

The following operators may be embedded in the *argument* for DBFIND *modes 1, 12, and 21*:

Conditional operators	
@	any variable number of alphanumeric characters
?	any single alphanumeric character
#	any single numeric character

Retrieval operators	
>=	greater than or equal to
<=	less than or equal to
<>	not equal to

Boolean operators	
SQL Notation	
AND	and
OR	or
NOT	and not
Infix Notation	
+	and
/	or
-	and not

Special Operators (for successive DBFIND calls)	
&	ANDs backup SI-subset with active SI-subset, replaces active with result, deletes backup
	ORs backup with active, moves result into active, deletes backup
!&	AND NOTs backup with active, moves result into active, does not change backup
/	moves copy of active into backup
\	moves backup into active, deletes backup
^	swaps the active and backup
[]	replaces backup with active, erase active
[*]	performs a projection
@@	rewinds virtual SI-chain; returns entry count in <i>mode 1</i>

❖ In *modes 100, 200, 300, 400 and 500*, the *argument* value is ignored.

The *argument* can contain ASCII or binary values for all modes, except for *mode 11* and *mode 22* which can only contain binary values. In *mode 13*, the *argument* is ignored.

For *mode* 11 and *mode* 22, the *argument* can only contain binary values. The *argument* length must be equal to twice the length of the index entry. Two binary values are passed in the *argument* array. The first binary value is the lower limit of a range, and the second binary value is the upper limit. Low-values in the first binary value and High-values in the second binary value would qualify all records.

Keywords If DBFIND is called against a keyworded SI-path and the length of the specified keyword exceeds the *keyword length* configured for the SI-path, the specified keyword is truncated to the keyword length and matching is done on the truncated value.

Argument If DBFIND is called in *mode* 1 or 21 and the specified *argument* is not the full SI-key
Terminator value, either the *buffer* must be padded with spaces or the *argument* value must be terminated by:

- for alphanumeric fields with *arguments* not surrounded with << >>, a single SPACE followed by a single @
- for alphanumeric fields with *arguments* not surrounded with << >>, a single @ (used for partial-key retrieval)
- for alphanumeric fields with the *argument* surrounded with << >>, up to two @s.
- for numeric fields, a single SPACE
- for *arguments* beginning with the tilde (~), the semi-colon (;)
- for *arguments* ending with a] or Boolean operator, a single SPACE or an @

For DBFIND *mode* 12 or 23, the *argument* must be terminated with a semi-colon (;).

Booleans Boolean operations between multiple values may be specified in a DBFIND *argument* in SQL Notation or Infix Notation. For example in *mode* 12:

~A@ or B@ or >=P@;	SQL Notation
~A@,B@,>=P@;	Infix Notation

These locate all entries that begin with "A" or "B", or that begin with "P" or an alphabetically higher letter.

Refinement The special operators for managing the active and backup SI-subsets may be used alone or in combination in the *argument*; if used alone, DBFIND will manage the SI-subsets but not select any new entries. Alternately, the special operator(s) may be used to prefix any *argument* value and both the selection and SI-subset management operations will be performed in the same intrinsic call.

Data types For DBFIND *modes* 1, 12, and 21, ASCII numbers may be specified for most numeric items (data types I, J, P, R, E, and Z) if prefixed with ==, >=, <=, or <> or, if appropriate, a - (negative sign)--the == operator simply converts an ASCII value specified to binary format for comparison.

For concatenated SI-keys that contain SI-subkeys of mixed data types (alphanumeric and numeric) there are many different ways to specify data. Refer to the **Finding entries in a concatenated SI-key** paragraph in [Section 4](#).

When performing relational access, the *argument* must be specified in ASCII. Values are automatically converted to binary for comparison with binary data values (do not prefix the argument with the **==** conversion operator).

For data types P and Z, SUPERDEX's DBFIND treats unsigned and positive values equivalently. For data type P, the sign is held in the last nibble (4 bits); for type Z, the sign is over-punched in the last byte.

Real numbers (items of data type R) may include embedded decimal points (**.**), exponential signs (**E**), and positive (**+**) and negative (**-**) signs.

Restrictions

A few restrictions exist in *arguments* that may be specified:

- for range searches, the first value specified (start point) must be less than the second value (end point)
- for range searches (as well as searches that use values that start with the **<=**, **>=**, or **<>** operators), a **?** or **#** embedded in a value is not recognized as an operator but as a regular character
- the **@** wildcard may only be specified as the last character in a value, unless the *argument* value is surrounded with the **<<** **>>** operators. Any characters that follow an **@** are ignored except when performing a range or not-equal-to retrieval
- exclusions (values preceded by **<>**) must be the last value in the *argument*

Data type Z

Additional restrictions exist in *arguments* that may be specified for SI-keys of data type Z:

- a **?** or **@** embedded in a value is not recognized as an operator but as a literal character. As the SI-key is numeric, this could result in "**ILLEGAL ASCII DIGITS**" in COBOL or as condition word 17.
- leading zeroes must be specified unless the *argument* is prefaced with the operator **==**, **<=**, **>=**, **<>** or relational access is being used.
- *mode 1nn*, *2nn*, *3nn*, *4nn*, or *5nn* must be used to qualify entries when using a partial key value for both simple and concatenated SI-keys
- for concatenated SI-keys, *mode 1nn*, *2nn*, *3nn*, *4nn*, or *5nn* must be used to qualify entries if the full SI-key value is specified
- for concatenated SI-keys that contain SI-subkeys of mixed data types and for which the first SI-subkey is numeric, a numeric value may be specified in *mode 1*, *12*, *21*, or *23* for the first SI-subkey only. Characters specified after the first SI-subkey will cause unpredictable results. Use *mode 1nn*, *2nn*, *3nn*, *4nn*, or *5nn* to qualify partial or full concatenated SI-keys

Effect of DBFIND on the SI-pointer and current path

The following table summarizes the effects of DBFIND on the SI-pointer and current SI-path:

<i>mode</i>	<i>condition word = 0</i>	<i>condition word <> 0</i>
1	before entry	current path not set**
12	before entry	current path not set**
21	before entry	current path not set**
23	before entry	current path not set**
100*	before first entry	current path not set
1nn	before entry	if <i>cw</i> = 17, before next entry
200*	before first entry	current path not set
2nn	before entry	if <i>cw</i> = 17, before next entry
300*	before first entry	current path not set
3nn	before entry	if <i>cw</i> = 17, before next entry
400	after last entry	current path not set
4nn	after entry	if <i>cw</i> = 17, after previous entry
500	after last entry	current path not set
5nn	after entry	if <i>cw</i> = 17, after previous entry

- * may be followed by either DBGET *mode* 5 or 15 or *mode* 6 or 16 and will start at either the beginning or end of the dataset, respectively, as compatible with IMAGE
- ** if using relational access, the current path is retained

DBGET intrinsic

❖ Remember that only SUPERDEX requirements are documented. For complete information on the Third-Party Indexing Interface External Specifications, see *Appendix A*.

Modes 5 and 6 after a SUPERDEX DBFIND return entries on a virtual SI-chain rather than a physical IMAGE chain. Also, DBGET *mode 5 and 6* work the same in both master and detail sets (in IMAGE, *mode 5 and 6* DBGETs against a master set traverse the synonym chain, and are rarely used).

Modes 15 and 16 are also available. These *modes* operate the same as *modes 5 and 6*, except they continue to retrieve entries even after the SI-keys no longer match the *argument*, all the way to the end or beginning of the set. Effectively, they perform greater-than-or-equal to and less-than-or-equal to retrievals, respectively.

Also, the `! list` construct is available for reading SI-indices only, for greater efficiency.

If the next- and previous-record numbers in the chain are required, set the JCW **SISSETLINK** to 1, regardless of the mode called. SUPERDEX will place the next- and previous-record numbers into the status array.

Refer to the *Retrieving entries with DBGET* chapter of the *Programming* section for further discussion.

Syntax DBGET (*base, dset, mode, status, list, buffer, argument*)

Parameters

Base Same as IMAGE

Dset Same as IMAGE--sets may be specified by name or number.

If accessing an independent SI-path, this parameter should be left blank or set to **200**.

Mode *Modes 5 and 6* work on SI-paths, and *modes 15 and 16* permit retrieval of entries in addition to those qualified by the previous DBFIND.

Mode 4 *Mode 4* continues to function as in IMAGE.

If the SI-path in the DBFIND is a one-to-one relationship index (simple index SI-path or concatenated SI-path) and no relational accessing was executed, this mode, in conjunction with modes 5 and 6, functions the same as IMAGE. If relational access was executed, it is not possible to reposition in the virtual SI-chain using DBGET mode 4. Modes 11, 12, and 13 should be used to reposition.

Mode 5 *Mode 5* continues to function as in IMAGE if the DBGET is performed on an IMAGE path.

Mode 5 against an SI-path returns all entries qualified by the previous DBFIND in ascending sorted sequential order if index access was used. Once all qualifying entries have been returned, the condition word is set to 15 ("END OF CHAIN").

Mode 6 *Mode 6* continues to function as in IMAGE if the DBGET is performed on an IMAGE path.

Mode 6 against an SI-path returns all entries qualified by the previous DBFIND in descending sorted sequential order if indexed access is used. Once all qualifying entries have been returned, the condition word is set to 14 ("BEGINNING OF CHAIN").

Mode 11 Resets the pointer for the virtual SI-chain to the beginning or the end of the qualified list. No current record is defined, and no buffer is filled after a positioning DBGET. DBGET mode 5 or 6 must be used to read a record.

Mode 12 Moves the pointer forward "*n*" indexes where "*n*" is a 32-bit unsigned integer value passed in the *argument* parameter. No current record is defined, and no buffer is filled after a positioning DBGET.

Mode 13 Moves the pointer backwards "*n*" indexes where "*n*" is a 32-bit unsigned integer value passed in the *argument* parameter. No current record is defined, and no buffer is filled after a positioning DBGET.

Mode 15 Same as *mode 5*, but continues to retrieve entries in ascending sorted sequential order even after the SI-keys no longer match the *argument* (greater-than-or-equal-to retrieval).

Mode 16 Same as *mode 6*, but continues to retrieve entries in descending sorted sequential order even after the SI-keys no longer match the *argument* (less-than-or-equal-to retrieval).

Status When using the SUPERDEX *modes* against an SI-path, only the first four words of the *status* array contain accurate information, unless the JCW **SISSETLINK** has been set to 1.

If the JCW **SISSETLINK** has been set to 1, words 7-8 and 9-10 will be set to the previous- and next-record numbers on the chain.

List

Same as IMAGE, but additional considerations for subsequent DBUPDATE and DBDELETE exist, and a new *list* is available.

If the **SIEXTLEN** JCW was used to configure a concatenated SI-key with more than four non-contiguous SI-subkeys, it is required that all items that were not explicitly referenced when defining the SI-key be included in the *list* in the order in which they appear in the dataset before calling DBUPDATE or DBDELETE.

! list

The **! list** returns the SI-index rather than the data entry (with a maximum of a 2K-word buffer). This is much faster than any other *list* construct because SUPERDEX needs to access only an SI-dataset.

Several functions can be accomplished with the **! list** by reading only the SI-index rather than the entire data record. For example:

- pattern matching and character validation
- intermediate storage of an SI-index to reset the SI-pointer after changing to another SI-path
- checking for the existence of common SI-indices in two datasets, by alternately reading SI-paths from each set

If the corresponding data entry is needed for any SI-index (to DBUPDATE or DBDELETE the data entry, for example) use DBGET *mode* 1 (reread current entry) with a *list* other than **!** to read it.

❖ **The **! list** is never seen by IMAGE, so the current *list* before the **! list** was specified will still be active (i.e. the *** list** may still be used as usual). In reading an independent SI-path, only the **! list** is allowed.**

SI-counter

If using the **! list**, an *SI-counter* may optionally be appended as a numeric literal to specify the number of SI-indices that should be returned by a single DBGET call, in the format:

!SI-counter

The *SI-counter* is terminated by **;** or blank

The number of indexes returned will be contained in STATUS words 5-6.

❖ **Make sure that the Buffer size specified is large enough to accommodate the number of indexes being returned.**

❖ **The maximum buffer size returned is 2K-words. If the requested number of indexes requires a buffer larger than 2K-words, an error -52 ("Bad List Parameter") will occur.**

Buffer Same as IMAGE.

Argument Ignored for *modes* 5, 6, 15, and 16.

Effect of DBGET on the SI-pointer and current path

The following table summarizes the effects of DBGET on the SI-pointer:

<i>mode/relationship</i>	<i>condition word = 0</i>	<i>condition word <> 0</i>
5 or 15*	before next entry	if <i>cw</i> = 15, no change
6 or 16*	after previous entry	if <i>cw</i> = 14, no change
other unique	on entry	no change
other non-unique	no change	no change

* if switching from *mode* 5 or 15 to *mode* 6 or 16, the same entry is not returned twice

DBINFO intrinsic

❖ Remember that only SUPERDEX requirements are documented. For complete information on the Third-Party Indexing Interface External Specifications, see *Appendix A*.

With the TurboIMAGE Third-Party Indexing Interface there are several additional DBINFO modes. Only these modes, in the 800 range, are described.

Parameters

All of the parameters are the same, except for the qualifier and return buffer layouts. These are the only parameters described for the Third-Party Indexing modes.

❖ The buffer layouts are described in 16-bit half-word lengths, as with standard TurboIMAGE.

Mode 801 Describes the Third-Party product enabled on the database.

Qualifier ignored.

Buffer

<i>word</i>	<i>description</i>
1-20	Product name (SUPERDEX Version (n.n))
21-25	Version number
26-27	CALENDAR date format of SUPERDEX Version
28-29	CLOCK time format of SUPERDEX Version

Mode 802 Returns the number of external and internal index files for this database.

Qualifier ignored.

Buffer

<i>word</i>	<i>description</i>
1	number of external files (may be 0).
2	number of internal files (may be 0).
3-10	Reserved for future use.

Mode 803 Indicates if SUPERDEX is enabled on the database.

Qualifier ignored.

Buffer

<i>word</i>	<i>description</i>
1	Binary 1 if enabled, 0 in not enabled.

Mode 811 Describes the access available for the data item or Third-Party search item.

Qualifier

<i>word</i>	<i>description</i>
1-8	TurboIMAGE data item name or number, or a Third-Party search item name or number.
9-16	Dataset name or number (Optional: If no dataset is to be specified, words 9-16 must equal spaces.)



For Independent paths, the dataset number should be set to 200. This is true for any DBINFO mode 8##.

Buffer

<i>word</i>	<i>description</i>
1	TurboIMAGE Item number or Third-Party search item number. If the <i>qualifier</i> is a TurboIMAGE item and a Third-Party search item, this value will always equal the TurboIMAGE item number.

Mode 812 Describes a Third-Party search item.

Qualifier

<i>word</i>	<i>description</i>
1-8	TurboIMAGE data item name or number, or a Third-Party search item name or number.
9-16	Dataset name or number (Optional: If no dataset is to be specified, words 9-16 must equal spaces).

Buffer

<i>word</i>	<i>description</i>
1-8	Third-Party search item name.
9	Data Type. Valid types (I,J,K,E,R,U,X,Z,P). Blank if a concatenated index.
10	IMAGE sub-item length. Blank if a concatenated index.
11	IMAGE sub-item count. Blank if a concatenated index.
12	0 if IMAGE item only, 1 if SUPERDEX item only, 2 if both
13	reserved for future use.

Mode 813 Describes all items in a database, including all Third-Party search items. Externals are the same as DBINFO *mode* 103.

Mode 814 Describes all items in a specific dataset, including all Third-Party search items. Externals are the same as DBINFO *mode* 104.

Mode 821 Identifies all datasets which contain the specified Third-Party key and the type of access allowed.

Qualifier

<i>word</i>	<i>description</i>
1-8	Third-Party search item name or number.

Buffer

<i>word</i>	<i>description</i>
-------------	--------------------

1	Number of datasets which contain the Third-Party key (<i>n</i>).
2 - (<i>n</i> +1)	IMAGE dataset numbers, including the access allowed by the DBOPEN password.

Mode 831 Identifies all SUPERDEX SI-paths. This DBINFO is used to list all Third-Party keys that can be called with DBFIND *mode* 1. Since all SUPERDEX SI-paths meet this criteria, all SI-paths are returned.

Qualifier

<i>word</i>	<i>description</i>
1-8	Dataset name or number.

Buffer

<i>word</i>	<i>description</i>
1	Number of Third-Party keys(<i>n</i>).
2 - (<i>n</i> +1)	Third-Party key numbers ($\geq 10,000$).

Mode 832 Identifies all SUPERDEX SI-paths. This DBINFO is used to list all Third-Party keys that can be called with DBFIND *mode* 12. Since all SUPERDEX SI-paths meet this criteria, all SI-paths are returned.

Qualifier

<i>word</i>	<i>description</i>
1-8	Dataset name or number.

Buffer

<i>word</i>	<i>description</i>
1	Number of Third-Party keys(<i>n</i>).
2 - (<i>n</i> +1)	Third-Party key numbers ($\geq 10,000$).

Mode 833 Describes a Third-Party key in detail.

Qualifier

<i>word</i>	<i>description</i>
1-8	Dataset name or number.
9-16	Third-Party key name or number.

Buffer

<i>word</i>	<i>description</i>
1	Third-Party key number ($\geq 10,000$).
2	Type of Key. "B" followed by a space which means can be called with DBFIND <i>mode 1</i> or <i>mode 12</i> .. If CORE SUPERDEX product level, then type of key will be "G"
3	SI-Index length in bytes (length includes the SI-Extension, if SICOGNOS JCW = 1, length does not include SI-Extension.) SI-Extension = detail set: 4 bytes or master set: search item length in bytes.
4	IMAGE set number if keyworded SI-path.
5	IMAGE item number of SI-subkey-1 if keyworded.
6	IMAGE item number of master search item if a master dataset.
7	IMAGE item length if keyworded.
8	0 if case not sensitive, 1 if case sensitive.
9	0 if not keyworded, 1 if keyworded.
10	0 if no keyword exclude path, 1 if keyword exclude path exists for database.
11	0 if blanks not indexed, 1 if blanks are indexed.
12	0 if not using Native Language Support, 1 if using NLS.
13-15	Always 0.
16-22	Reserved for future use.
23	0 if not grouped, 1 if standard group, 2 if SUPERGROUP.
24	Date format indicator: 0 if not a date format (ASK or PowerHouse date). +1 if OLD ASK date (pre-version 8.0) +2 if NEW ASK date (version 8.0 or later) +3 if PowerHouse date +10 if day of month is included +100 if month is included +1000 if year is included +10000 if century is included
25	Average number of keywords, 0 if not keyworded.
26	Minimum length of keyword, 0 if not keyworded.
27	SI-Index dataset (0-7).
28	Number of SI-Subkeys (1-4).
29	SI-Subkey-1 IMAGE item number.
30	SI-Subkey-1 Byte Offset (one relative; If SICOGNOS jcw = 1, then zero relative).
31	SI-Subkey-1 Length is bytes.
32	SI-Subkey-1 IMAGE item type (I, J, K, E, R, U, X, Z, P).
33	SI-Subkey-1 IMAGE item length.
34	SI-Subkey-1 IMAGE item compound count.
35-40	SI-Subkey-2 Information.
41-46	SI-Subkey-3 Information.
47-52	SI-Subkey-4 Information.

Mode 834 Identifies all IMAGE Items grouped with the specified Third-party key.

Qualifier

<i>word</i>	<i>description</i>
1-8	Dataset name or number.
9-16	Third-party key name or number.

Buffer

<i>word</i>	<i>description</i>
1	Number of IMAGE Items in the group(<i>n</i>).
2	IMAGE set number of first item in the group.
3	SI-subkey-1 IMAGE item number of first item in the group.
4	IMAGE set number of second item in the group.
5	SI-Subkey-1 IMAGE item number of second item in the group.
6 ...	<< until end of all items (up to 32) in the group >>

DBOPEN intrinsic

The DBOPEN intrinsic is the same as in IMAGE, although there are now +100 modes.

❖ Remember that only SUPERDEX requirements are documented. For complete information on the Third-Party Indexing Interface External Specifications, see *Appendix A*.

Syntax DBOPEN (*base,password,mode,status*)

Parameters

Base Same as IMAGE.

Password Same as IMAGE.

Mode **+100** Adding 100 to the existing DBOPEN modes opens the database but disables the Third-party Indexing retrievals. This affects only the DBFIND and DBGET intrinsics.

❖ There is a new JCW, IMAGETPI, that can be set to 100 before any DBOPEN is executed. This allows the same functionality as the +100 DBOPEN modes, without requiring modification to the source. This should only be used to disable SUPERDEX lookups during tests.

Status Same as current IMAGE.

DBPUTIX intrinsic

DBPUTIX is a new SUPERDEX intrinsic used to explicitly add SI-indices into B-trees. Its counterpart is the new DBDELIX intrinsic, which explicitly deletes SI-indices.

The DBPUTIX intrinsic accesses only the appropriate SI-dataset and is used to maintain independent indices. It also provides a method for adding custom SI-indices in addition to those maintained automatically by DBPUT.

Refer to the *Adding, updating, and deleting entries* chapter of the *Programming* section for further discussion.

Syntax **DBPUTIX** (*base,dset,mode,status,item,buffer*)

The DBPUTIX intrinsic is syntactically similar to DBPUT except that the *list* parameter is replaced by an *item* parameter and the *buffer* parameter contains the full SI-index, including the SI-extension.

Parameters

Base	The <i>base-ID</i> (same as DBPUT).
Dset	Name or number of the dataset in which the corresponding data entry exists. If accessing an independent SI-path, this parameter should be left blank or set to 200 .
Mode	An integer with the value 1 .
Status	Only the condition word is set.
Item	The name of the SI-path in whose B-tree to add the specified SI-index.
Buffer	The full SI-index value including the SI-extension, which is: <ul style="list-style-type: none">■ the search field value for a master dataset■ the relative record number for a detail dataset■ a suitable user-defined value for independent SI-paths

Error handling

Since SUPERDEX uses standard IMAGE messages to report all errors and exceptional conditions, the same messages that are used by DBPUT are displayed for DBPUTIX.

SITRANSLATE intrinsic

SITRANSLATE is a procedure to translate input from Infix notation to the Reverse Polish Notation.

Infix operators are **+** (AND), **-** (AND NOT), and **,** (*comma*) (OR). The input string is evaluated from left to right, with standard precedence between operators (AND before NOT, etc...). Parentheses can be used to change the order of evaluation. A **-** (AND NOT) after an opening parenthesis is translated into a unitary NOT (see example 5, following).

Operands containing imbedded blanks must be delimited by double quotes ("**"**"). The input string must be terminated by a blank.

The **OPERATOR** parameter allows one infix operator to be supplied programmatically. This is useful in applications prompting the user only for operands.

Parameters

Operator	(<i>byte</i>)	Values allowed are "+", "-", ",", and blank (no operator supplied)
Input	(<i>byte array</i>)	Input string to be translated, terminated by blank, in infix notation.
Output	(<i>byte array</i>)	Output string of translation, terminated by blank, in Reverse Polish Notation.
Error	(<i>integer</i>)	0 = upon successful operation. 1 = illegal OPERATOR parameter 2 = parentheses do not match 3 = input not complete (# of operators/operands do not match) 4 = operator stack overflow (Too many parentheses) 5 = too many operators (Maximum of 3) 6 = output overflow (Limited to 255 characters)

Examples

- 1) OPERATOR = *blank*
INPUT = NEW-"NEW YORK"
OUTPUT = [NEW][NEW YORK]!&
- 2) OPERATOR = *blank*
INPUT = (COMB+BLIND), (HOLD+DRILL)-FASTEN@
OUTPUT = [COMB][BLIND]&[HOLD][DRILL]&|-[FASTEN@]!&
- 3) OPERATOR = +
INPUT = "LOS ANGELES"
OUTPUT = [LOS ANGELES]&
- 4) OPERATOR = *blank*
INPUT = +"LOS ANGELES"
OUTPUT = [LOS ANGELES]&
- 5) OPERATOR = *blank*
INPUT = (-"NEW YORK")-"LOS ANGELES"
OUTPUT = [NEW YORK]![LOS ANGELES]!&

SIUSER procedure



WARNING!!!!

Because SUPERDEX is called from within IMAGE, SIUSER routines will be called in "PRIVILEGE" mode and "SET-CRITICAL" mode.

THIS MEANS ANY FAILURE WITHIN A SIUSER ROUTINE WILL CAUSE A SYSTEM FAILURE!!

Always verify your SIUSER routines thoroughly, and check all possible statuses and error conditions that may occur in the SIUSER routine.

SIUSER is an optional user-written procedure that is invoked by SUPERDEX to compute one or more custom SI-indices for entries whenever DBPUT, DBUPDATE, or DBDELETE is called and from the SIMAINT utility. It is useful for establishing SI-indices that cannot be composed of dataset fields as they are represented but which can be calculated using values in the data entry, as well as for SI-indices that require:

- more than four non-contiguous SI-subkeys
- date conversion
- reordering
- upshifting
- stripping
- other parsing

The SIUSER procedure is written by the user and installed in an XL that is named XL<base name>.group.account where the base.group.account match the database for the SIUSER. For example, for the database OEDB.ORDER.PARTS, the SIUSER would be located in the XL named XLOEDB.ORDER.PARTS.



This new XL MUST be located in a Group-Account with PM capability. If PM capability cannot be installed on the Group-Account where the database exists, the XL can be :FILE equated. For example, the XL can be located in SUPERDEX.SYS, and referenced with a :FILE equation:

:FILE XLOEDB.ORDER.PARTS=XLOEDB.SUPERDEX.SYS

If multiple custom SI-paths have been configured, it is necessary to add conditional statements into the SIUSER procedure to specify which statements are executed for which SI-path.

Syntax

SIUSER (*base,dset,item,buffer,index*)

Parameters

The first four parameters are supplied by the program; the *index* value is returned by SIUSER.

Base	The <i>Base-id</i> (returned by DBOPEN).
Dset	The name or number of the dataset in which the corresponding data entry exists. When called from DBPUT, DBUPDATE, or DBDELETE, its format is the same as specified in those intrinsics. If called from the SIMAINT program, the dataset name is used.
Item	The name of the SI-path in which to add the specified SI-index.
Buffer	The full data entry (@ <i>list</i>) used in the DBPUT or DBDELETE.
Index	This is an output parameter only--its value is returned by SIUSER. The first word contains a count of the number of SI-indices to be created, followed by their values in the length defined for the SI-path. Up to 16 indices may be returned. To cause SUPERDEX not to generate any indices for an entry, specify 0 in the first word of this parameter. SUPERDEX will automatically add the appropriate SI-extension to form the SI-index by appending either the entry's IMAGE search field value (if a master dataset) or its relative record number (if a detail).

DBPUT, DBUPDATE, and DBDELETE

SI-indices generated by the SIUSER procedure are automatically maintained by DBPUT, DBUPDATE, and DBDELETE.

SIMAINT utility

The SIMAINT utility, when generating or reorganizing SI-indices, calls SIUSER from the specified XL in the RUN command and automatically generates the corresponding indices.

If an SIUSER procedure is written to access an SI-path from within the procedure, the SI-path that is accessed from within the SIUSER procedure must have been configured in a previous run of SIMAINT and may not be changed in the current run.

SECTION 6:**Maintenance and utilities**

This section discusses the various maintenance considerations for databases that contain SI-paths. It also reviews the various utility programs that may be used with SUPERDEX'ed databases.

Chapter 1 Database maintenance considerations

Function Includes tables listing the various operations that require maintenance of SI-paths and the type of maintenance required.

Chapter 2 SIMAINT utility

Function Used to create, reorganize and delete SI-paths. It also describes the DBLOAD, LIST, SCHEMA, and STRUCT options, and running SIMAINT in batch.

Chapter 3 SUPERDEX utility

Function This program is used to maintain and reorganize SI-paths in an on-line full screen environment.

Chapter 4 SIPATH utility

Function Will display all IMAGE keys and SUPERDEX SI-path information in one concise display.

Chapter 5 SITEST and SIREPAIR utilities

Function Used to check the integrity of B-trees and their correspondence to the data entries they represent, and to repair the B-trees.

Chapter 6 SICOUNT utility

Function This utility will display the exact compression information on SI-paths.

Chapter 7 SITRACE facility

Function Available with SUPERDEX. This facility will trace all user IMAGE intrinsic calls, along with the SUPERDEX IMAGE intrinsic calls.

Chapter 8 SIDRIVER utility

Function IMAGE Intrinsic processor, similar to DBDRIVER.PUB.SYS from Hewlett-Packard.

CHAPTER 1:**Database maintenance considerations**

SI-indices are maintained automatically by DBPUT, DBDELETE, and DBUPDATE intrinsics. However, because SI-indices reference data entries by search field value or relative record number, certain database maintenance functions will cause the SI-indices to lose synchronization with the data entries they map and require reorganization.

Examples of this are as follows:

- database maintenance tasks, such as reorganizing a detail dataset
- database structural modifications, such as changing the length of an item used as an SI-key
- modifications to the KWEXCLUD keyword exclusion file

The following tables list the various database conditions that can require SI-path maintenance.

Error and exceptional conditions that can require SI-path maintenance

<i>Type</i>	<i>Description</i>
<i>Condition</i>	Overflow of insufficient capacity in SI-dataset
<i>Action</i>	Reorganize all SI-paths in the affected SI-dataset using <i>dataset/R</i> . If unsuccessful, run SIMAINT,DBLOAD
<i>Condition</i>	Failed dataset erase
<i>Action</i>	Erase the dataset with a utility or delete any remaining entries using QUERY.PUB.SYS, then reorganize all SI-paths related to the dataset. If unsuccessful, run SIMAINT,DBLOAD.

Database maintenance tasks that can require SI-path maintenance

<i>Type</i>	<i>Description</i>
<i>Function Action</i>	Renaming an item or dataset. Run SIMAINT (without the STRUCT entry-point) and press RETURN at the DATASET> prompt.
<i>Function Action</i>	Any function that causes entries in a detail dataset to move to different physical locations, such as a detail dataset reorganization. Reorganize all SI-paths in the related dataset.
<i>Function Action</i>	Database DBUNLOAD/DBLOAD. Run SIMAINT,DBLOAD
<i>Function Action</i>	A database erase (such as with DBUTIL.PUB.SYS). The SUPERDEX configuration will also be erased. MUST redefine all SI-paths.
<i>Function Action</i>	Changing the length or data type of an item used in an SI-key. Run SIMAINT,STRUCT. For items which the SI-key length may not be specified (numeric), the SI-key length is readjusted automatically; otherwise, the SI-key length is unchanged unless it would exceed the item length, in which case the SI-key length is reduced.
<i>Function Action</i>	Changing the name of an item used in a SI-key. Run SIMAINT and press RETURN at the DATASET> prompt.
<i>Function Action</i>	Changing the name of a dataset that has at least one SI-path. Run SIMAINT and press RETURN at the DATASET> prompt.
<i>Function Action</i>	Deleting all SI-datasets. Delete the SI-item.

Other conditions that can require SI-path maintenance

<i>Type</i>	<i>Description</i>
<i>Function Action</i>	Modifying the keyword exclude file (KWEXCLUD). Reorganize the KWEXCLUDE SI-path and all of the keyworded SI-paths.

Redefining and reorganizing SI-paths

To redefine an SI-path or all the SI-paths related to a dataset, use the SIMAINT utility to Delete (**/D**) and then redefine the configuration.

To reorganize a selected SI-path or all the SI-paths related to a dataset, use the Reorganize (**/R**) option of the SIMAINT utility on the a specified SI-path or dataset.

To reorganize all SI-paths for a dataset or database while also regenerating the SI-definitions, run SIMAINT with the DBLOAD entry point.

DBGENERAL interface

Bradmark's general-purpose database maintenance utility, **DBGENERAL**, automatically performs the appropriate maintenance tasks against SUPERDEX structures whenever necessary (DBGENERAL version 6.0 and later).

For example, when erasing a dataset using DBGENERAL option 4.4, the corresponding SI-indices are automatically removed; when reorganizing a detail dataset using option 3.6, the corresponding SI-indices are automatically reorganized; and when renaming a dataset that has SI-paths using options 5.3 and 5.6, the internal SUPERDEX definitions are automatically updated.

Refer to the **DBGENERAL User Manual** for more information.

CHAPTER 2:**SIMAINT utility**

The SIMAINT utility is used for both configuring new SI-paths and maintaining existing SI-paths. It also contains functions for displaying the SUPERDEX configuration for a database and regenerating the SI-definitions following structural changes to a database. These options are invoked by one of the following entry points:

- (none) permits SI-paths to be added, reorganized, and deleted
- **DBLOAD** reorganizes all the SI-paths for a dataset or database; regenerates the SI-definitions
- **LIST** lists all the SI-paths configured for a database
- **SCHEMA** generates a job stream to configure SI-paths based on current configuration
- **SHARED** permits shared access for SI-paths to be added, reorganized, and deleted
- **STRUCT** adjusts the SI-definitions to compensate for changes to database structure

Although new SI-paths may be defined concurrently with maintenance of existing SI-paths, only the SIMAINT functions for reorganizing and deleting SI-paths are discussed here--refer to the *Configuration/Establishing indices* section for discussions about and examples of using SIMAINT to configure new SI-paths and group existing SI-paths.

Access requirements

Before running SIMAINT, make sure:

- you have exclusive access to the database (except when using the LIST, SCHEMA and SHARED options)
- you are logged on as the database creator
- you are logged into the group and account in which the database resides

It is also recommended for performance reasons that you:

- disable ILR
- disable logging
- do not run SIMAINT with `;XL="???", ;LIB=G` or `;LIB=P`

Input rules

These rules govern SIMAINT input:

- all input may be in upper- or lower-case
- `?` displays structural help (sets and items)
- `\` flushes the current response and re-prompts
- lengths are reported and specified in words, not bytes (unless otherwise specified), and it is necessary to convert for alphanumeric (data types U and X) items (e.g. X20 = 10 words).

SIEXTLEN JCW for special concatenated SI-keys

If you have configured SI-keys that contain more than four non-contiguous items by utilizing the **SIEXTLEN JCW**, it is required that this JCW be set before running SIMAINT with the STRUCT or SCHEMA entry points. To do so:

```
:SETJCW SIEXTLEN=1
```

Invoking SIMAINT

To invoke SIMAINT:

```
:RUN SIMAINT.SUPERDEX.SYS
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993
```

Specifying the database

Specify the name of a database, as shown:

```
Database > OEDB
```

Specifying datasets

After specifying the database name, a list of the datasets that contain SI-paths is displayed:

```
SI-paths exist for the following sets:
  - blank -
ORDER-HEADERS
ORDER-LINES
CUSTOMERS
Enter name of set to be modified or new name
```

SIMAINT can be run against datasets that already have related SI-paths or those for which SI-paths have not yet been configured. If a dataset already contains SI-paths, they are displayed.

The - blank - set references independent indexes. See [Section 3: Configuring/Establishing SI-indexes](#).

Enter the name of a manual master or detail dataset in the current database that contains SI-paths, optionally followed by one of the following suffixes:

- /D** Delete all dataset's related SI-paths
- /R** Reorganize all dataset's related SI-paths

This command instructs SIMAINT to reorganize all the SI-paths related to the CUSTOMERS dataset:

```
Dataset > CUSTOMERS/R
```

❖ **The SI-dataset suffix (/1 through /7) with which the dataset was optionally defined is automatically retained and may not be overridden. To reassign a dataset's SI-indices to the root SI-dataset or any other SI-dataset, it is necessary to delete and redefine all related SI-paths.**

Specifying SI-paths

All SI-paths related to the specified dataset and their attributes are displayed, as shown:

```
Dataset >customers
The following SI-paths and Items are defined:
CUSTOMER-NAME-KW/K  CUSTOMER-NAME      L =   4
CUSTOMER-NAME      CUSTOMER-NAME      L =  15
ADDRESS1-CITY-KW/K ADDRESS-1          L =   4
ADDRESS1-CITY-KW/K CITY                        L =   4
Enter SI-path with option /D /R /G or new name
```

Enter the name of one of the SI-paths shown appended by one of the following suffixes:

- /D** Delete specified SI-path
- /G** Group specified SI-path (refer to the *Configuration/Establishing indices* section)
- /R** Reorganize specified SI-path

Reorganizing SI-paths

SI-paths should be reorganized periodically to maintain optimum performance, and must be reorganized after certain database maintenance operations. SI-paths may be selected individually or by related dataset. Alternately, all the SI-paths for a dataset or database may be reorganized by running SIMAINT with the DBLOAD option, which also regenerates the SI-definitions.

Refer to the tables near the beginning of this section for complete details on what database maintenance tasks require SI-path reorganization.

For super-grouped SI-paths, the SI-indices for all SI-paths configured in the super-group are automatically reorganized whenever any SI-path in the super-group is reorganized.

This example specifies the reorganization of all the SI-paths related to the CUSTOMERS dataset:

```
Dataset > CUSTOMERS/R
Dataset >
```

SI-paths may alternately be specified individually; for keyworded SI-paths, the average number of indices may be changed, as shown:

```
Dataset > CUSTOMERS
SI-path > CUSTOMER-NAME-KW/R
Enter average number of indices per entry > 4
SI-path >
```

In this example, the keyworded SI-path *CUSTOMER-NAME-KW* is being reorganized by suffixing it with */R*, and the average number of indices per entry is being changed to 4.

Deleting SI-paths

SI-paths may be selected for deletion individually or by related dataset.

For super-grouped SI-paths, all SI-paths configured in the super-group are automatically deleted whenever any SI-path in the super-group is deleted.

In this example, all the SI-paths related to the ORDER-LINES dataset are deleted:

```
Dataset > ORDER-LINES/D
Dataset >
```

In this example, the SI-path *PART-ORDER* is being deleted by suffixing it with */D*:

```
DATASET > ORDER-LINES
The following SI-paths and Items are defined:
ORDER-PART          ORDER-NUMBER      L = 2          PART-NUMBER      L = 7
PART-ORDER          PART-NUMBER       L = 7          ORDER-NUMBER     L = 2
PART-DESC/K         PART-DESCRIPTION  L = 4          PART-NUMBER      L = 7
Enter SI-path with option /D /R /G or new name
SI-path > PART-ORDER/D
SI-path >
```

DBLOAD Entry Point

The **DBLOAD** entry point is used to reorganize all the SI-paths for a dataset or database. This entry point is recommended following an operation in which data entries are relocated in the database, such as a DBUNLOAD and DBLOAD. It is also useful for giving users the ability to safely and easily reorganize existing SI-paths--especially untrained users and those in turnkey environments--since only the base name needs to be specified.

Additionally, SIMAINT,DBLOAD regenerates the SUPERDEX configuration for a database, and is required after certain database maintenance functions. A complete list of database maintenance operations that require the use of SIMAINT,DBLOAD appears in tables near the beginning of this section.

In this example, all the SI-paths for the OEDB database are reorganized:

```

:RUN SIMAINT.SUPERDEX.SYS,DBLOAD
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993
Database >OEDB
DBLOAD:      1246 B-tree records deleted      CPU 0:00:21.4 Elapsed 0:00:35
SI-paths exist for the following sets:
  - blank -
CUSTOMERS
ORDER-HEADERS
ORDER-LINES
Enter name of set to be modified or new name
Dataset > RETURN
Processing SI-path KWEXCLUDE      OF
Processing SI-path CUSTOMER-NAME-KW OF CUSTOMERS      # of Ent: 1002
  Input:      1002 records      100 %      CPU 0:00:02.9 Elapsed 0:00:04
  Sort:      2997 Indices      CPU 0:00:00.0 Elapsed 0:00:00
  Output:      2983 Indices      100 %      CPU 0:00:02.4 Elapsed 0:00:03
Processing SI-path CUSTOMER-NAME OF CUSTOMERS      # of Ent: 1002
  Input:      1002 records      100 %      CPU 0:00:00.9 Elapsed 0:00:01
  Sort:      1001 Indices      CPU 0:00:00.0 Elapsed 0:00:00
  Output:      1001 Indices      100 %      CPU 0:00:01.5 Elapsed 0:00:02
Processing SI-path ADDRESS1-CITY-KW OF CUSTOMERS      # of Ent: 1002
  Input:      1002 records      100 %      CPU 0:00:04.0 Elapsed 0:00:04
  Sort:      4348 Indices      CPU 0:00:00.0 Elapsed 0:00:00
  Output:      4317 Indices      100 %      CPU 0:00:03.3 Elapsed 0:00:04
Processing SI-path CUSTOMER-NUMBER OF ORDER-HEADERS      # of Ent: 2620
  Input:      2620 records      100 %      CPU 0:00:02.1 Elapsed 0:00:03
  Sort:      2620 Indices      CPU 0:00:00.0 Elapsed 0:00:00
  Output:      2620 Indices      100 %      CPU 0:00:01.8 Elapsed 0:00:03
Processing SI-path ORDER-TYPE OF ORDER-HEADERS      # of Ent: 2620
  Input:      2620 records      100 %      CPU 0:00:01.9 Elapsed 0:00:02
  Sort:      2620 Indices      CPU 0:00:00.0 Elapsed 0:00:00
  Output:      2620 Indices      100 %      CPU 0:00:01.7 Elapsed 0:00:03
Processing SI-path ORDER-PART OF ORDER-LINES      # of Ent: 9272
  Input:      9272 records      100 %      CPU 0:00:07.9 Elapsed 0:00:12
  Sort:      9272 Indices      CPU 0:00:00.0 Elapsed 0:00:00
  Output:      9272 Indices      100 %      CPU 0:00:09.7 Elapsed 0:00:13

```

Processing SI-path PART-ORDER		OF ORDER-LINES		# of Ent: 9272
Input:	9272 records	100 %	CPU 0:00:08.0	Elapsed 0:00:09
Sort:	9272 Indices		CPU 0:00:00.0	Elapsed 0:00:00
Output:	9272 Indices	100 %	CPU 0:00:10.2	Elapsed 0:00:13
Processing SI-path PART-DESC		OF ORDER-LINES		# of Ent: 9272
Input:	9272 records	100 %	CPU 0:00:31.4	Elapsed 0:00:40
Sort:	32641 Indices		CPU 0:00:00.2	Elapsed 0:00:01
Output:	32622 Indices	100 %	CPU 0:00:37.5	Elapsed 0:00:42

LIST Entry Point

When the **LIST** entry point is used, SIMAINT opens the specified database in shared access mode and lists all its SI-paths:

```

:RUN SIMAINT.SUPERDEX.SYS,LIST
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database >OEDB
The following SI-paths and items are defined:
Dataset      SI-path                      Items/Lengths

 10001  KWEXCLUDE                      4
CUSTOMERS
 10002  CUSTOMER-NAME-KW/K  CUSTOMER-NAME          4
 10003  CUSTOMER-NAME          CUSTOMER-NAME          15
 10004  ADDRESS1-CITY-KW/K  ADDRESS-1              4
 10004  ADDRESS1-CITY-KW/K  CITY                   4
ORDER-HEADERS
 10005  CUSTOMER-NUMBER      CUSTOMER-NUMBER        2
 10006  ORDER-TYPE          ORDER-TYPE             1
ORDER-LINES
 10007  ORDER-PART          ORDER-NUMBER           2    PART-NUMBER           7
 10008  PART-ORDER          PART-NUMBER            7    ORDER-NUMBER          2
 10009  PART-DESC/K          PART-DESCRIPTION       4    PART-NUMBER           7

END OF PROGRAM

```

A number following a set name, preceded by a slash, indicates the number of the SI-dataset that contains the SI-indices for that dataset. The number prior to the SI-path name is the SI-path number. The letter following an SI-path name, preceded by a slash, indicates whether the SI-path is Keyworded or has Blank values indexed. SI-path names that appear more than once are grouped or super-grouped together.

SCHEMA Entry Point

The SCHEMA entry point causes SIMAINT to generate a job stream based on the current SI-definitions which can be used to completely redefine the SUPERDEX configuration for a database. This is useful for duplicating the SUPERDEX configuration from one database to another, as well as recovering the SI-definitions should the root SI-dataset be erased or corrupted.

When running SIMAINT,SCHEMA, you are prompted for the base name and name of a file in which to write the job stream as shown:

```

:RUN SIMAINT.SUPERDEX.SYS,SCHEMA
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database >OEDB
SIMAINT Job File Generation

Enter NEW filename for SIMAINT job > OEDBSI

:END OF PROGRAM

```

If the name of an existing file is specified, you are prompted for whether or not to overwrite it; otherwise, a new file is created and the job stream is written into it, as shown:

```

!JOB MGR.SUPERDEX,DEMO40
!COMMENT Patch PASSWORD if necessary !
!COMMENT TimeStamp : THU, DEC 10, 1992, 10:14 AM
!RUN SIMAINT.SUPERDEX.SYS
<Base >OEDB
<Dataset >
<SI-path >KWEXCLUDE
<KW length >4
<SI-path >//
<Dataset >CUSTOMERS
<SI-path >CUSTOMER-NAME-KW/K
<Item 1 >CUSTOMER-NAME
<Item len=15 - KW length >4
<Min chars per KW >1
<Average keywords per entry >5
<Item 2 >//
<SI-path >CUSTOMER-NAME
<Item len=15 - Shorter len >-30
<Offset (in bytes) >1
<Item 2 >//
<SI-path >ADDRESS1-CITY-KW/K
<Item 1 >ADDRESS-1
<Item len=13 - KW length >4
<Min chars per KW >1
<Average keywords per entry >6

```

```

<Item 2 >//
<SI-path >ADDRESS1-CITY-KW/G
<Item 1 >CITY
<SI-path >//
<Dataset >ORDER-HEADERS
<SI-path >CUSTOMER-NUMBER
<Item 2 >//
<SI-path >ORDER-TYPE
<Item len=1 - Shorter len >-2
<Offset (in bytes) >1
<Item 2 >//
<SI-path >//
<Dataset >ORDER-LINES
<SI-path >ORDER-PART
<Item 1 >ORDER-NUMBER
<Item 2 >PART-NUMBER
<Item len=7 - Shorter len >-14
<Offset (in bytes) >1
<Item 3 >//
<SI-path >PART-ORDER
<Item 1 >PART-NUMBER
<Item len=7 - Shorter len >-14
<Offset (in bytes) >1
<Item 2 >ORDER-NUMBER
<Item 3 >//
<SI-path >PART-DESC/K
<Item 1 >PART-DESCRIPTION
<Item len=13 - KW length >4
<Min chars per KW >2
<Average keywords per entry >16
<Item 2 >PART-NUMBER
<Item len=7 - Shorter len >-14
<Offset (in bytes) >1
<Item 3 >//
<SI-path >//
<Dataset >//
<The following input is processed only by SIMAINT,
if no SI Dataset(s) exist, and is ignored otherwise.
It denotes default capacity and should be edited if necessary.
Capacity SI dataset >//
!EOJ

```

Before streaming the job stream, it may be necessary to change one or more of the following parameters:

- add user, group, and/or account passwords if necessary
- specify a capacity for each SI-dataset rather than retaining the current capacity

-
- ❖ When creating the job file, if the SIEXTLEN JCW is set, a corresponding **:SETJCW** command is included in the job stream. If the JCW is not set and should be (based on the SUPERDEX configuration for the specified database), the message *****WARNING - JCW SIEXTLEN NOT DEFINED***** is displayed. If this warning is displayed, set the JCW and rerun SIMAINT,SCHEMA.
-

STRUCT Entry Point

The **STRUCT** entry point causes SIMAINT to compare the SUPERDEX configuration with the database structure and correct any inconsistencies found. It is used after making certain structural changes to a database, and is automatically invoked by DBGGENERAL when necessary. It should, however, not be run after performing a DBUNLOAD/DBLOAD--the SIMAINT program should be run with the DBLOAD option instead.

Refer to the tables near the beginning of this section for a list of maintenance functions that require the use of the STRUCT option (you will note that the STRUCT option is never used after renaming sets or items).

This example shows SIMAINT,STRUCT being used after changing the length of an item used as an SI-key:

```

:RUN SIMAINT.SUPERDEX.SYS,STRUCT
SIMAINT Version 4.2
Demo (all options) valid until THU, JAN 21, 1993

Database > OEDE
Dataset > RETURN

```

Running SIMAINT in batch

SIMAINT can be run in batch, and uses dialog similar to that when running it on-line. The method for creating a job stream by which to run SIMAINT in batch is to run SIMAINT with the ,SCHEMA entry-point.

The discrepancies between on-line and batch use are:

- some prompts are asked in batch at all times, while on-line the prompts are asked depending on previous answers
- a line containing only a **SPACE** is represented in batch by a blank line
- a line containing only a **RETURN** (which is normally specified in a batch job as a blank line) should be represented by a line containing a double slash (//) in the first two character positions

SIMAINT will QUIT (not TERMINATE) normally upon encountering any error in batch, permitting testing of the system JCW.

CHAPTER 3:

SUPERDEX utility

The SUPERDEX program was created to provide a menu interface to the SUPERDEX environment. It is a full screen menu driven interface to most programs associated with the SUPERDEX product, as well as providing an on-line full screen interface to SIMAINT.

This interface takes advantage of VPLUS screens to allow the maintenance of SUPERDEX SI-paths by selecting options through the menu instead of the command driven process of SIMAINT. The information is requested in a user-friendly logical manner, and edits all input to ensure adherence to the rules of the SUPERDEX product. SUPERDEX will capture all the information for maintaining SI-paths and optionally execute on-line or create a file that will apply the modifications in batch.

Access requirements

Before running SUPERDEX, make sure:

- you have shared (DBOPEN *mode 5*) access to the database
- you are logged into the account in which the database resides
- you have a database password that allows read-access to **ALL** sets and items in the database (can use the creator password).

Invoking SUPERDEX

The SUPERDEX program can be executed from anywhere within the account where the database exists. If the database is located in a group other than the log on group, you can set up a file equation to redirect access to the database, or you can qualify the group name when selecting the database in the SUPERDEX program.

The SUPERDEX program is located in SUPERDEX.SYS. To invoke SUPERDEX:

```
:RUN SUPERDEX.SUPERDEX.SYS
```

Additionally, there is a large internal table that can be saved using the "**SAVE**" option in the "**;INFO=**" parameter. For example:

```
:RUN SUPERDEX.SUPERDEX.SYS ;INFO="SAVE=file.group"
```

If "**SAVE**" is entered without the "**=*file.group***", the internal table will be stored to a file named SITABLE. "**SAVE=TEMP.PUB**" will store the table in the file named TEMP in the group PUB.

Function Key Operation

The function keys are used extensively in the SUPERDEX program to allow for easy movement through the screens. The values of each function key will change between screens, and depending on available options, on the same screen.

There are two function keys that are always the same throughout the SUPERDEX program; **F1** and **F8**. **F1** is the HELP function key and can be accessed at any time on any screen. **F8** is the EXIT function key. Pressing this key will return control back to the Main Menu screen, or if at the Main Menu screen, it will exit the SUPERDEX program.

Main Menu

```

SUPERDEX VERSION 4.0   BRADMARK TECHNOLOGIES, INC. (c)BRADMARK 1987, 1993

  1  INSTALLATION
  2  PATH MAINTENANCE

1.1  Install SUPERDEX
      On a Database

2.1  Maintain SI-Path(s)
2.2  > Generate Path File
2.3  > Reorg All SI-Paths

  3  DIAGNOSTICS
  4  OTHER FEATURES

3.1  General Path Info
3.2  Detail Path Info
3.3  Test SI-Path(s)
3.4  > Repair SI-Path(s)
3.5  Compression Info
3.6  > Structural Change

4.1  > Command Driven Maint.

Please enter the number
of the desired selection: __

                                     > Exclusive Access

```

The Main Menu provides access to most of the programs in the SUPERDEX environment. Each can be accessed by entering the number associated with the particular program (1.1 through 4.1). Upon completion of the selected program, control will return back to this point. Refer to the other chapters in this section for more information on each of the programs.

There are only two function keys available on this screen. **F1** accesses the HELP Screen and **F8** will EXIT the SUPERDEX program.

Once the option has been keyed, press the **ENTER** key.

Base Menu

APR 03, 1993	SUPERDEX BASE MENU	10:53 AM
Please enter base name: _____ Password(Dflt=Creator): _____		
SUPERDEX (C) Bradmark Technologies, Inc. 1993		

This screen is used to define the database and password options.

First, you must enter the name of the database you wish to maintain. This can include the group designation if the database is located in a group different from the log on group.

Second is the database password, which defaults to the database creator password. This password must be a password that gives read-access to ALL of the IMAGE items and datasets.

There are only two function keys available on this screen. **F1** access the HELP Screen and **F8** will return to the Main Menu.

Once the database information has been keyed, press the **ENTER** key to advance to the Dataset Menu.

The last column is a display-only field used to display the 16 character dataset name. There is always a dataset displayed as ****SPECIAL****. This dataset does not exist in the database but is used to define all Independent SI-paths and the Keyword Exclude path. The actual SI datasets (SI,SI1,SI2,etc.) will be displayed, but they can not be selected.

There are several function keys available from this screen:

Dataset Menu Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2	DEFINE SI-PATHS	Advances to the Path Menu. (Only displayed if datasets have been selected)
F3	BEGIN AGAIN	Exits the Dataset Menu and returns to the Base Menu.
F4		(Not Defined)
F5		(Not Defined)
F6	NEXT SCREEN	Displays the next screen of datasets
F7	PREV SCREEN	Displays the previous screen of datasets
F8	EXIT	Exits back to the Main Menu.

After the datasets have been selected, press the **ENTER** key. To advance to the Path Screen or the Special Path Screen, press the **F2** function key.

❖ **If the SI-path is deleted and it is a SuperGroup SI-path, the related Master or Detail dataset(s) will be searched to ensure that all parts of the SuperGrouped SI-path are deleted.**

The `Path Type:` fields follow. Each one of these options operate differently. As with the `Action:` options, to select one enter any character, such as **X**, in the respective field.

The first option for SI-path types is `Group`. This option can be selected for existing SI-paths or for new SI-paths. If the SI-path exists, it is used to group other IMAGE items to the SI-path and no other options are valid. If the SI-path is new, it is used to allow grouping multiple IMAGE items in the new SI-path in one pass. In this case, either the `Keyword` or `Blank` option can additionally be selected, but only one and no others.

The second option is `SuperGroup` and can only be selected for new SI-paths in Detail datasets. The SI-path does not need to already exist in the Master dataset unless it should be a concatenated SuperGroup. In this case, first add the concatenated SI-path for the Master. It is not necessary to designate an SI-path for a Master dataset as SuperGroup. If the SI-path does exist as a concatenated SI-path in a related Master, SUPERDEX will enforce that the items in the concatenated Master SI-path exist in the Detail, except for SI-SUBKEY-1.

The next two options, `Keyword` and `Blank`, are mutually exclusive and can be used in conjunction with the `Group` option. Again, placing any character (e.g. **X**) in the field will select the option, and these options can only be selected for new SI-paths.

The last option is the `Custom` option. This option can not be used in conjunction with any of the other `Path Type:` options and can only be used on new SI-paths. This option is used to mark that the indices will be created in an SIUSER subroutine (see [Section 2](#) for more information). Before a custom SI-path is defined, the SIUSER subroutine must already exist for the path to be populated.

There are several function keys available from this screen:

Path Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2	NEXT DATASET	Displays the next dataset in the selection order. (May advance to the Special Path Screen)
F3	PREVIOUS DATASET	Displays the previous dataset in the selection order. (May return to the Special Path Screen)
F4	DATASET SCREEN	Returns to the Dataset Menu.
F5	EXECUTE SCREEN	Advances to the Execute Screen.
F6		(Not Defined)
F7		(Not Defined)
F8	EXIT	Exits back to the Main Menu.

Once all of the data has been keyed, press the **ENTER** key to advance to the `Item Screen` or the `Custom Path Screen`.

Special Path Screen

```

APR 03, 1993                SUPERDEX SPECIAL PATH SCREEN                10:53 AM

Dataset: _____ Selection Order: __
SI-Path Name: _____ Action: _ Delete
Path Length (2/63): __      Use NLS-Sorting (Y/N): _
                              (answer only if * is displayed)

Existing Paths:
Len Path                      Len Path                      Len Path
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
* ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX    * ## XXXXXXXXXXXXXXXXXXXX
SUPERDEX (C) Bradmark Technologies, Inc. 1993

```

This screen is the processing screen for ****SPECIAL**** SI-paths. ****SPECIAL**** SI-paths include the Independent SI-paths and the keyword exclude SI-path, KWEXCLUDE (see [Section 2](#) for more information). It is used to create or delete ****SPECIAL**** SI-paths.

The first two fields are display only fields. They display ****SPECIAL**** for the dataset name and the selection order value entered from the Dataset Menu.

The next field is the SI-path Name:. This field is used to enter the name of either an existing SI-path or a new SI-path. If the name entered is an existing SI-path, only the Action: Delete option can be selected. If the name entered is not an existing SI-path the Action: option is not available. If you wish to modify an existing SI-path, it must first be deleted and then added back.

The Action: Delete option is next. As stated earlier, this option is only available for existing SI-paths. To mark the path for deletion, key any character, such as **X**, in the field.

The next two fields, Path Length and NLS-sorting, are for new SI-paths. The Path Length is used to define the entire length, in words, for the new path, and can be any numeric value between 2 and 63. The NLS-Sorting option will only accept a value if an * (asterisk) is displayed next to the field. The asterisk will only be displayed if the database has been marked for a Native-Language other than English. If the NLS collating sequence should be used to sort the indices, type **Y**, otherwise the default **N** will be used. **Only select NLS if the independent path will contain alphanumeric data.** If it will contain only binary numeric values, use the default of **N**.

Next, there are three columns used to display currently defined SI-paths. The first field in the column is the asterisk used to flag that an existing SI-path that has NLS sorting turned on. The next field is the **Len** field and is used to display the defined length of the SI-path. The Path field is used to display the SI-path name.

The function keys available for this screen are:

Special Path Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2	NEXT DATASET	Displays the next dataset in the selection order (Will advance to the Path Screen).
F3	PREVIOUS DATASET	Display the previous dataset in the selection order (Will return to the Path Screen).
F4	DATASET SCREEN	Returns to the Dataset Menu.
F5	EXECUTE SCREEN	Advances to the Execute Screen.
F6		(Not Defined)
F7		(Not Defined)
F8	EXIT	Exits back to the Main Menu.

Once all of the data has been keyed, press the **ENTER** key. The Special Path Screen will still be displayed for more entry.

Custom Path Screen

```

APR 03, 1993                SUPERDEX CUSTOM PATH SCREEN                10:53 AM

Dataset: _____ Selection Order: __
SI-path Name: _____
SI-KEY Length (2/63): __
Average number of indexes per record (1/16): __
Use NLS-sorting (Y/N): _
(answer only if "*" is displayed)

SUPERDEX (C) Bradmark Technologies, Inc. 1993

```

The Custom Path Screen is used to enter the information for a custom path (see [Section 3](#) for more information). Since a custom path does not have any IMAGE items, this screen is used instead of the Item Screen used for regular SI-paths.

The first three fields are display only fields. They display the current dataset being processed, the selection order value entered from the Dataset Menu, and the name of a new SI-path from the Path Screen.

The next field, SI-KEY Length is used to define the length, in words, for the new path not including the SI-Extension (see [Appendix B](#) for more information), and can be any numeric value between 2 and 63, inclusive.

Next is the Average number of indexes per record field. This is used to calculate the proper capacity for the corresponding SI dataset and to calculate the proper size for sorting the indexes. The value can be any numeric value from 1 to 16.

Finally, the NLS-Sorting option will only be accepted if an * (asterisk) is displayed next to the field. The asterisk will only be displayed if the database has been marked for a Native-Language other than English. If the NLS collating sequence should be used to sort the indices, type **Y**, otherwise the default **N** will be used. **Only select NLS if the custom path will contain alphanumeric data.** If it will contain only binary numeric values, use the default of **N**.

There are three function keys available from this screen:

Custom Path Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2		(Not Defined)
F3	PATH SCREEN	Returns to the Path Screen, without accepting the data currently displayed on the screen.
F4		(Not Defined)
F5		(Not Defined)
F6		(Not Defined)
F7		(Not Defined)
F8	EXIT	Exits back to the Main Menu.

Once all of the data has been keyed, press the **ENTER** key. The process will then return to the Path Screen.

Below is a description of all of the fields:

Path Display Screen Fields	
Label	Description
Dataset:	Current dataset being accessed
Selection Order:	Selection order of dataset
SI-path Name:	Name of the SI-path
Path Type:	Full description of the type of path
Key Length:	Byte length of the SI-key
Number of Keys:	Total number of IMAGE items from the dataset that are included in this SI-path
Min Key Word Length:	Minimum number of characters required to create a keyworded index
Index Length:	Total length of the SI-index (includes the SI-key and SI-extension) in bytes
Avg No Key Words:	The defined average number of keywords per record
S1, S2, S3, S4	These designations are used to identify the items that are included in a concatenated SI-path. S1 is for SI-subkey-1, S2 for SI-subkey-2, etc. If the SI-path is not concatenated, none of the designations will be displayed.
Items	Names of the IMAGE items included in the SI-path
Type	The IMAGE item type (as returned by DBINFO mode 102)
Length	The IMAGE item length (as returned by DBINFO mode 102)

The function keys available from this screen are:

Path Display Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2		(Not Defined)
F3		(Not Defined)
F4		(Not Defined)
F5		(Not Defined)
F6	NEXT SCREEN	Displays the next screen of items
F7	PREV SCREEN	Displays the previous screen of items
F8		(Not Defined)

To return to the Path Screen, press the **ENTER** key.

For existing SI-paths, the items already included will be displayed with an appropriate letter designation. These **CANNOT** be removed or changed. Only new items can be marked.

If creating a SuperGroup SI-path for a Detail dataset only one item can be selected (using **A**). If there are no related Master dataset(s) with the same SI-path, the process will advance to the Related Masters Menu.

The function keys available from this screen are:

ITEM Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2	DEFINE SI-ITEMS	Advances to the Item Definition Screen.
F3	PATH SCREEN	Returns to the Path Screen, without updating the item data
F4		(Not Defined)
F5		(Not Defined)
F6	NEXT SCREEN	Displays the next screen of items
F7	PREV SCREEN	Displays the previous screen of items
F8		(Not Defined)

To update the data, press the **ENTER** key. If the items are not numeric, you must also press the **F2** key to define the item specific information.

Item Definition Screen

APR 03, 1993	SUPERDEX ITEM DEFINITION SCREEN	10:53 AM
Dataset: _____	Selection Order: __	
SI-path Name: _____		
Path Type: _____		
Item: _____	Item Selection Order: __	
Item length (in bytes):	_____	
Starting Position (in bytes):	_____	
Minimum Number of Characters: per Keyword (Keyword only)	__	(1 - 4)
Average Number of Keywords: per record (Keyword only)	__	(1 - 16)
SUPERDEX (C) Bradmark Technologies, Inc. 1993		

The Item Definition Screen is used to define detailed information about the IMAGE items that were selected on the previous screen. This screen will request information for fields based on the SI-path type and the IMAGE item type. For example, a numeric item will not be displayed because the IMAGE definition must be used, or if the SI-path is not a keyword path, the fields marked Keyword only do not apply.

The first four fields are all display only and are passed from previous screens. The Dataset: is the name of the dataset being accessed, the Selection Order: is the selection order keyed from the Dataset Menu, the SI-path Name: is the name of the SI-path being accessed, and the Path Type: is the description of the type of the SI-path as defined on the Path Screen.

The next two fields, Item: and Item Selection Order: are also display fields (Passed from the Item Screen).

The Item length: field is used to enter the length of the SI-subkey in bytes. By default, for non-keyworded SI-paths, the length will be set to the length of the IMAGE item. For keyworded SI-paths, the length will be defaulted to eight (8) bytes.

The Starting Position: is used to define the position of where the index should begin. For non-keyworded SI-paths, the value can be any starting position (relative to 1), as long as the starting position is not longer than the length of the IMAGE item. For keyworded SI-paths, this value will be forced to one (1).

The next two fields are only valid for keyworded SI-paths. The first, Minimum Number of Characters:, is used to define the minimum number of characters that must be in the word before it will be indexed. The default is two (2). This means that single character words, such as "a" will not be indexed. The valid values are 1 through 4, inclusive.

The Average Number of Keywords: is used to calculate the size of the sort file and the worst case capacity for the SI datasets. The default is set to the size of the field, divided by seven (7), always rounded up. For example, a CUSTOMER-NAME item defined in IMAGE as an X30 would have a default of 5 (30 divided by 7 equals 4.29, then rounded up to 5). A 60 character description item would be defaulted to 9 (60 divided by 7 equals 8.57, rounded to 9). This value must be 16 or less.

The function keys available from this screen are:

Item Definition Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2		(Not Defined)
F3	ITEM SCREEN	Returns to the Item Screen, without updating the item data
F4		(Not Defined)
F5		(Not Defined)
F6		(Not Defined)
F7		(Not Defined)
F8	EXIT	Exits back to the Main Menu

To update the data, press the **ENTER** key. Once the data has been updated, the process will return to the Path Screen.


```

APR 03, 1993          SUPERDEX GROUPED ITEM OFFSET SCREEN          10:53 AM

Dataset: _____ Selection Order: _
SI-path Name: _____
SI-Subkey1: _____ Key Length: ____ Start: ____

STARTING          STARTING          STARTING
POSITION          ITEM            POSITION          ITEM            POSITION          ITEM

--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____
--  _____  --  _____  --  _____

SUPERDEX (C) Bradmark Technologies, Inc. 1993
    
```

The Grouped Item Offset Screen is used to accept the offset (starting position in characters) of each grouped item. This screen is only displayed for those grouped paths that are of an alphanumeric type.

Each grouped item has a default offset of 1, which can be either accepted or overridden. The maximum offset for any item cannot exceed the length of the item itself.

For existing paths that are having additional items grouped to them, this screen will only display the new items being grouped (currently existing grouped items will not be displayed).

There are two function keys available for this screen.

Grouped Item Offset Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2		(Not Defined)
F3		(Not Defined)
F4		(Not Defined)
F5		(Not Defined)
F6		(Not Defined)
F7		(Not Defined)
F8	EXIT	Exits back to the Main Menu.

Once all of the data has been keyed, press the **ENTER** key . The process will then return to the PATH Screen.

Execute Menu

```

APR 03, 1993                SUPERDEX EXECUTE SCREEN                10:53 AM

_ (1) Execute On-line
  (2) Execute in Batch
  (3) Save Job file [Enter file name]: _____
                                     file.group

_ (1) Modify Index Structure and Populate Indices
  (2) Modify Index Structure, only

Job Name:      _____

User Name:    _____ Password: _____

Group Name:   _____ Password: _____

Account Name: XXXXXXXXX Password: _____

SUPERDEX (C) Bradmark Technologies, Inc. 1993

```

This screen is used to select how the modification process should be executed. By default, the process will execute the SUPERDEX maintenance program, SIMAINT, as a son process using the information captured.

The first field is used to select the execution option. **1**, the default, runs the process immediately on-line, **2** streams the process for batch immediately, and **3** is used to save the information entered in a job file that can be streamed at a later time.

If **2** was chosen, a job file named **SIBCHFLE** will be created and streamed. If for some reason SIMAINT does not complete successfully (e.g. the SI dataset did not have enough free space to add the new index), this file will not be automatically purged. Otherwise, it will be purged by the job stream itself. If **3** was chosen, the second field is used to enter a valid MPE file name (with the group optional) in which to save the job file.

The third field selects whether the process will only define the new SUPERDEX index structure, or will also populate the SUPERDEX indices for the new structure. The default is to populate the indices during the definition execution. To not process the indices, simply enter **2** and at some later point, it will be necessary to reorganize the newly created indices.

The last group of fields are used to define the logon necessary for the job file. The Job Name : is an optional name that defaults to **SIBATCH** and can be modified.

The **User Name :** and **Password :** are required fields for logon. The name field will default to the current logon **USER** name and can be modified. The password field is used to enter the user's logon password, if it exists.

The **Group Name :** is used to enter the group name of where the database is located. It is set to either the current logon **GROUP** or to the qualified database group name from the **Database Menu**, if it was specified. Again, the **Password :** field is used to enter the group logon password, if necessary.

The **Account Name :** is displayed, and can not be modified. If an **ACCOUNT** password exists, it must be entered in the **Password :** field.

❖ **If this is the first SUPERDEX index structural change made to the database, there will be questions about capacities for the SI dataset(s) during the maintenance process. The processes will use the default values for the capacities calculated by SIMAINT. These are always the worst case calculation based on the capacity of the user dataset(s). Therefore, none of the options selected will fail because of a full SI dataset. Once the process has completed, an adjustment to the capacity of the SI dataset(s) may be done.**

The function keys available from this screen are:

Path Display Screen Function Keys		
Key	Label	Description
F1	HELP	Displays the Help Screen.
F2	DATA SCREEN	Returns to the Dataset Menu for addition maintenance
F3		(Not Defined)
F4		(Not Defined)
F5		(Not Defined)
F6		(Not Defined)
F7		(Not Defined)
F8	EXIT	Returns back to the Main Menu and does not save the entered information

To update the data, press the **ENTER** key. If on-line was chosen, the process will run the maintenance program, **SIMAINT**, displaying the progress as it processes, and then return to the **Main Menu**. If the selection to execute the process in batch immediately was chosen, the job file will be streamed and the process will return to the **Main Menu**. If the choice was made to save the job file, the information will be saved and the process will return to the **Main Menu**.

CHAPTER 4:**SIPATH utility**

SIPATH is a program that will display the IMAGE keys and chains, along with the SUPERDEX SIPATH information. This information is given in a concise manner and is complete.

To use SIPATH, simply:

```
:RUN SIPATH.SUPERDEX.SYS
```

At this point the first window will be displayed. This window prompts for the database name and password. Simply type in the database name, press **RETURN**, type in the password and press **RETURN**. The database name can be fully defined, including the GROUP and the ACCOUNT, and the password will not be displayed for security reasons.



At any time during the execution of SIPATH, the user can press the F1 function key for on-line help. When the F1 is pressed, help information about the current window will be displayed.

```
|
|
|-----|
| SIPATH Version 4.2 |
|-----|
|
|__Base Information__|
|-----|
| Base Name:         |
|                    |
| Password :         |
|                    |
|-----|
|
|-----|
| Please Wait - Loading Base Information |
|-----|
|
```

While the database information is being loaded, the `PLEASE WAIT` message will be displayed. Once the database information is loaded into SIPATH, the datasets for the database will be displayed. The screen will contain three windows. The top window displays the SIPATH header information. The second window displays the selected database, dataset, and SI-path (at this time the dataset and SI-path have not been selected).

The third window is a scroll window of the valid datasets. A dataset name will be highlighted. The user can then use the up-arrow or down-arrow to change the highlighted dataset. The user can select the highlighted dataset by pressing the **RETURN** key. Pressing the **F8** (Cancel) key will return the user to the Base window.

```
|-----|
| SIPATH Version 4.2 |
|-----|

|_Information_ |
| Base Name : OEDB.DEMO40      Set Name : |
| Path Name : |
|-----|

|_Datasets_ |
| Dataset      Type |
| CUSTOMERS    M    |
| ORDER-HEADERS M    |
| ORDER-LINES  D    |
|-----|
```

When the user selects a dataset, the dataset name will be displayed in the second window, and a fourth scroll window of the valid SI-paths will be displayed next to the dataset scroll window.

The SI-paths window works the same as the Dataset window. An SI-path will be highlighted. The user can use the up-arrow or down-arrow to highlight different SI-paths, and then press **RETURN** to select the highlighted SI-path. Pressing the **F8** (Cancel) key will return the user to the Dataset scroll window.

```

|-----|
| SIPATH Version 4.2 |
|-----|

|__Information__|
| Base Name : OEDB.DEMO40          Set Name : CUSTOMERS |
| Path Name :                      |
|-----|

|__Datasets__| |__SI-paths__|
| Dataset      Type | | SIPath |
| CUSTOMERS    M   | | CUSTOMER-NAME-KW |
| ORDER-HEADERS M   | | CUSTOMER-NAME |
| ORDER-LINES  D   | | ADDRESS1-CITY-KW |
|-----| |-----|

```

When the user selects an SI-path the SI-path name will be displayed in the second window, and the SI-path detail information window will be displayed. This window displays all of the information about an SI-path, and pressing **F8** (CANCEL) will return the user to the SI-path scroll window.

```

SIPATH Version 4.2

Information
Base Name : OEDB.DEMO40          Set Name : CUSTOMERS
Path Name : CUSTOMER-NAME-KW

SI-path Detail
Name      : CUSTOMER-NAME-KW      Total length : 8
Number    : 10002                 Index Dataset: 0

Index Options:
Keyworded : Y   Ave. No. of KW: 5   Min. KW length : 1
Index Blanks: N   Grouped          : N   Date Type      :
NLS Enabled : N   Super Grouped    : N   Date Format     :

IMAGE Items:
Name      Number  Type      SUPERDEX
Offset   Length
Subkey 1: CUSTOMER-NAME    2      X30      1      8
Subkey 2:                   0
Subkey 3:                   0
Subkey 4:                   0
    
```

The information on the SI-path detail screen is broken down into three parts.

The top part is header information about the SI-path. It contains the name of the SI-path, the number of the SI-path (always greater-than-or-equal-to 10,000), the total length (does not include the SI-Pointer length), and which SI-Dataset is used for the indexes (0 - 7).

The middle of the window displays options defined on the SI-path. These include whether the SI-path is keyworded, and if so, the defined average number of keywords and the minimum keyword length. The second line displays whether empty fields, or blank indexes, should be indexed, whether the SI-path is a grouped path and, if the SI-path was defined as a date, what type of date (ASK, including whether it is the old format or new, or PowerHouse). The third line displays whether Native-Language Support (NLS) is enabled on the path, whether it is SuperGrouped, and if it is a date path, what the format of the date is. The date format displays what pieces of the date are selected. These are century (CC), year (YY), month (MM), and day (DD). The format can be CCYYMMDD or any combination.

The last part of the window displays IMAGE information about the items that make up the SI-path. There can be a maximum of four IMAGE items. If more than one is displayed, the path is a "concatenated" path. The name of the IMAGE item, along with its IMAGE item number and type is displayed. This information will match the definition as defined in the data base. The last two columns display the offset and length in characters of the IMAGE item used.

If the SI-path selected is a Grouped or SuperGrouped path, there will be more items to display than just the SI-subkeys. Function Key F2 can be pressed to display all of the IMAGE items that make up the Grouped or SuperGrouped path.

```

SIPATH Version 4.2

Information
Base Name : OEDB.DEMO40          Set Name : CUSTOMERS
Path Name : ADDRESS1-CITY-KW

SI-path Detail
Name      : ADDRESS1-CITY-KW      Total length : 8
Number    : 10004                Index Dataset: 0

Index Options:
Keyworded : Y   Ave. No. of KW: 6   Min. KW length : 1
Index Blanks: N   Grouped      : Y   Date Type      :

NL_Grouped Items
SetName      SetType  ItemName   ItemNo   ItemType
CUSTOMERS    M        ADDRESS-1   3        X26
CUSTOMERS    M        CITY        5        X10
Sub
Sub
Sub
Sub
    
```

The Grouped Items window displays the IMAGE items along with all the information about the item. It's IMAGE item number and type, and which dataset and dataset type the item is located in. If the SI-path is a Grouped SI-path, the dataset name will match the selected dataset in the Information Window. If the SI-path is a SuperGrouped SI-path, the dataset names will be different. Some will be from the master dataset and some from one or more detail datasets.

There can be up to 32 items joined together in a Grouped or SuperGrouped SI-path. This window is a scroll window and the user can use the up-arrow and down-arrow to move up and down the list of items included in the group.

CHAPTER 5:

SITEST and SIREPAIR utilities

The SITEST utility is used for checking the integrity of the SUPERDEX B-trees and their correspondence to the data entries they represent. Additionally, SITEST verifies the KWEXCLUDE keyword exclusion SI-path against the KWEXCLUD disk file.

SIREPAIR adds the capability to actually repair the SUPERDEX B-trees, so they will match the data. SIREPAIR will repair up to 1% of the indexes based on the number of entries in the dataset. There is an upper limit of one million indexes that can be repaired. This is because the process of repairing the indexes is slower than the process of reorganizing the indexes. If more than 1% of the indexes are corrupted, it shows major corruption on the B-tree and therefore the B-tree should be completely reorganized.

Inconsistencies are reported by an ASCII/octal/hex dump of the SI-indices in error. Also, each inconsistency indicates whether it is an SI-index with no corresponding entry or an entry with no corresponding SI-index.

Access requirements

Before running SITEST or SIREPAIR, make sure:

- you have shared (DBOPEN *mode 5*) access to the database for SITEST, or exclusive (DBOPEN *mode 1*) access to the database for SIREPAIR
- you are logged on as the database creator
- you are logged into the group and account in which the database resides

Because SITEST and SIREPAIR do extensive locking, it is recommended that they not be run during heavy user access.

The rules and processes of SIREPAIR are the same as SITEST. Therefore, first SITEST will be covered, then the differences between SITEST and SIREPAIR.

Invoking SITEST

To invoke SITEST:

```
:RUN SITEST.SUPERDEX.SYS
```

```
SITEST Version 4.2
```

Specifying the database

Specify the name of a SUPERDEX'ed database, as shown:

```
Database > OEEDB
```

Specifying datasets

Specify @ to diagnose all datasets, the name of a dataset that contains SI-paths, or **SPACE + RETURN** to diagnose a independent SI-path:

```
Dataset > @
```

Specifying SI-paths

If a value other than @ was specified for dataset, a prompt is issued to determine whether to diagnose a specific SI-path for the current dataset or all SI-paths for the dataset. Enter an SI-path name or @ to diagnose all SI-paths for the dataset:

```
SI Path > @
```

Specifying mode of operation

SITEST has two modes of operation:

Mode 1 checks the integrity of the B-tree itself and does not validate its relationship to the data entries.

Mode 2 checks the integrity of the B-tree (same as mode 1) and the correspondence of the SI-indices to the data entries they represent. Mode 2 is slower than mode 1 but performs a more thorough analysis.

```
Mode (1=Tree, 2=Full) >
```

Mode 1 processing

If mode 1 was selected, SITEST processes each B-tree in succession and displays the phase TREETEST while processing. Once the processing has completed, the number of SI-indices checked is displayed:

```

Mode (1=Tree, 2=Full) > 1
Processing SI-path KWEXCLUDE      OF
TreeTest          0 Indices          CPU 0:00:00.0 Elapsed 0:00:00
Processing SI-path CUSTOMER-NAME  OF CUSTOMERS
TreeTest         1003 Indices         CPU 0:00:00.3 Elapsed 0:00:01
Processing SI-path CUSTOMER-NAME-KW OF CUSTOMERS
TreeTest         3059 Indices         CPU 0:00:00.4 Elapsed 0:00:01
Processing SI-path ADDRESS1-CITY-KW OF CUSTOMERS
TreeTest         4418 Indices         CPU 0:00:00.5 Elapsed 0:00:01
Processing SI-path CUSTOMER-NUMBER OF ORDER-HEADERS
TreeTest         2620 Indices         CPU 0:00:00.3 Elapsed 0:00:00
Processing SI-path ORDER-TYPE     OF ORDER-HEADERS
TreeTest         2620 Indices         CPU 0:00:00.2 Elapsed 0:00:00
Processing SI-path ORDER-PART     OF ORDER-LINES
TreeTest         9272 Indices         CPU 0:00:01.7 Elapsed 0:00:03
Processing SI-path PART-ORDER     OF ORDER-LINES
TreeTest         9272 Indices         CPU 0:00:01.7 Elapsed 0:00:03

END OF PROGRAM
    
```

Mode 2 processing

If mode 2 was selected, SITEST processes each B-tree in succession and displays the phase TREETEST while performing the B-tree check. Following the TREETEST, the phases INPUT, SORT, and COMPARE are executed and displayed accordingly. Once the processing of each SI-path is completed, the number of SI-indices checked is displayed:

```

Mode (1=Tree, 2=Full) > 2
Processing SI-path KWEXCLUDE      of
TreeTest          0 Indices          CPU 0:00:00.0 Elapsed 0:00:00

Processing SI-path CUSTOMER-NAME  of CUSTOMERS
TreeTest         1004 Indices         CPU 0:00:00.2 Elapsed 0:00:00
Input            1004 Records         100 %   CPU 0:00:00.9 Elapsed 0:00:01
Sort             1004 Indices         CPU 0:00:00.0 Elapsed 0:00:00
Compare          0 Indices
    
```

```

*** Inconsistency: Keys from SI Chain vs. Keys from Dataset *****

SI Key #   Key Value (SI Chain or Dataset)                                DSet Key #

          052105 051524 020055 020116 047440 044516 042105 054040   957
           T E S T      -      N O      I N D E X
           54 45 53 54 20 2D 20 4E 4F 20 49 4E 44 45 58 20
          020040 020040 020040 020040 020040 020040 020040 000020
                                     .....
           20 20 20 20 20 20 20 20 20 20 20 20 00 10
          172107
           ... G
           F4 47

          957 052105 051524 020055 020116 047440 051105 041517 051104
           T E S T      -      N O      R E C O R D
           54 45 53 54 20 2D 20 4E 4F 20 52 45 43 4F 52 44
          020040 020040 020040 020040 020040 020040 020040 000041
                                     ... !
           20 20 20 20 20 20 20 20 20 20 20 20 00 21
          164216
           .....
           E8 8E

1004 Indices from SI Chain, 1004 Indices From Dataset Compared
2 ERROR(S) found

Processing SI-path CUSTOMER-NAME-KW of CUSTOMERS
TreeTest          3062 Indices          CPU 0:00:00.4 Elapsed 0:00:00
Input             1004 Records          100 % CPU 0:00:02.4 Elapsed 0:00:03
Sort              3077 Indices          CPU 0:00:00.0 Elapsed 0:00:00
Compare           3062 Indices          100 % CPU 0:00:01.4 Elapsed 0:00:02
Processing SI-path ADDRESS1-CITY-KW of CUSTOMERS
TreeTest          4418 Indices          CPU 0:00:00.5 Elapsed 0:00:01
Input             1004 Records          100 % CPU 0:00:03.7 Elapsed 0:00:04
Sort              4449 Indices          CPU 0:00:00.0 Elapsed 0:00:00
Compare           4418 Indices          100 % CPU 0:00:02.1 Elapsed 0:00:02
Processing SI-path ORDER-TYPE of ORDER-HEADERS
TreeTest          2620 Indices          CPU 0:00:00.2 Elapsed 0:00:00
Input             2620 Records          100 % CPU 0:00:02.2 Elapsed 0:00:02
Sort              2620 Indices          CPU 0:00:00.0 Elapsed 0:00:00
Compare           2620 Indices          100 % CPU 0:00:01.1 Elapsed 0:00:01
Processing SI-path ORDER-PART of ORDER-LINES
TreeTest          9272 Indices          CPU 0:00:01.5 Elapsed 0:00:02
Input             9272 Records          100 % CPU 0:00:08.6 Elapsed 0:00:09
Sort              9272 Indices          CPU 0:00:00.0 Elapsed 0:00:00
Compare           9272 Indices          100 % CPU 0:00:05.4 Elapsed 0:00:06

```

Processing	SI-path	PART-ORDER	of	ORDER-LINES		
TreeTest		9272 Indices			CPU 0:00:01.6	Elapsed 0:00:02
Input		9272 Records	100 %		CPU 0:00:08.9	Elapsed 0:00:10
Sort		9272 Indices			CPU 0:00:00.0	Elapsed 0:00:00
Compare		9272 Indices	100 %		CPU 0:00:05.8	Elapsed 0:00:06

For each inconsistency detected, the SI-index is displayed with a counter indicating the relative position in the B-tree. If the counter is on the left side of the output, it indicates an SI-index with no corresponding entry; if the counter is on the right side, there is an entry with no corresponding SI-index.

Running SITEST in batch

SITEST can be run in batch, and uses the same dialog as on-line. The method for creating a job stream by which to run SITEST in batch is to anticipate the on-line prompts and provide responses for them.

SITEST will QUIT (not TERMINATE) normally upon encountering any error in batch, permitting testing of the system JCW.

Invoking SIREPAIR

As stated earlier, while discussing SIREPAIR we will only identify the differences between SITEST and SIREPAIR..

To invoke SIREPAIR:

```
:RUN SIREPAIR.SUPERDEX.SYS  
  
SIREPAIR Version 4.2
```

Specifying Input

The input for database, datasets, and SI-paths are the same for SIREPAIR as they are for SITEST.

Specifying request before update

SIREPAIR will now prompt to see if the updating should be done immediately, or should the user specify that the SI-path should be repaired.

```
Prompt before repair (Y,N) >
```

After this, SIREPAIR will do the same processing as SITEST mode 2. After it displays the indexes that are corrupt, it will prompt the user if the previous question was answered yes.

```
Repair this path ? (Y,N) >
```

If the SI-path should be repaired, SIREPAIR will repair the reported inconsistencies between the indices and the data and then go on to the next SI-path.

CHAPTER 6:

SICOUNT utility

SICOUNT was created to provide the exact compression information in the B-trees. SUPERDEX compresses B-trees based on the full index values for a physical SI data record. This means all of the indexes in a given physical SI data record will be compressed the same.

SUPERDEX begins compressing with the left most word and will compress up to the whole length of an SI-INDEX, including the SI-EXTENSION. The default size of the SI-ITEM is set with this compression algorithm taken into consideration. It is possible to increase, or decrease, the compression by manually setting up the SI-ITEM and SI dataset structures, although this is not recommended. Depending on the data and the type of items in the SI-path, a smaller SI-ITEM can produce a greatly increased compression ratio.

❖ **Since there are several items to consider, contact Bradmark Technical Support before adjusting the structure of the SI-ITEM and SI datasets.**

SICOUNT will not only provide the information on the compression, but will also give the height of the B-trees, the number of leaf and tree records, and the number of indexes in the path.

Invoking SICOUNT

To invoke SICOUNT:

```
:RUN SICOUNT.SUPERDEX.SYS
```

Specifying the database

Specify the name of a SUPERDEX'ed database, as shown:

```
Database > OEDB
```


Specifying datasets

Specify @ to diagnose all datasets, the name of a dataset that contains SI-paths, or **SPACE + RETURN** to diagnose a independent SI-path:

```
Dataset > @
```

Specifying SI-paths

If a value other than @ was specified for dataset, a prompt is issued to determine whether to diagnose a specific SI-path for the current dataset or all SI-paths for the dataset. Enter an SI-path name or @ to diagnose all SI-paths for the dataset:

```
SI Path > @
```

Process

At this point SICOUNT will process the SI-paths selected and will display detailed information about each path:

```
KeyLength      =
BTree Height   =
# Tree Records =
# Leaf Records =
# Keys         =

LenPrefix #Records  #Keys Avg.Keys/Rec Max Keys/Rec Optimal Keys/Rec
                                CPU 0:00:00.0 Elapsed 0:00:00
```

Following is a table explaining all of the items displayed:

Label	Description
KeyLength	The total length of the SI-index, including the SI-extension, in words.
BTree Height	The height of the B-tree. This will define the maximum number of physical I-Os to qualify a record. If the B-tree records are in memory, there will be no physical I-Os.
# Tree Records	The total number of physical records that make up the B-tree.
Contiguous pairs	The number of physical records included in the B-tree, that are contiguous in the SI-dataset (Also called "Index Efficiency"). The percentage of efficiency is also reported.
# Leaf Records	The number of physical records that are located on the bottom of the tree.
# Keys	The total number of SI-indices in the B-tree.
LenPrefix	The number of words in the SI-index that are compressed. "0" is no compression. "3" is three words compressed in every index in the physical record.
#Records	The number of physical records that are compressed at the level.
#Keys	The number of SI-indices that are compressed at the level.
Avg Keys/Rec	The average number of SI-indices that are contained on one physical record.
Max Keys/Rec	The maximum number of SI-indices that are contained on one of the physical records.
Optimal Keys/Rec	The optimal number of SI-indices that can be contained on one physical record.

Here is an example of SICOUNT.

```

SICOUNT version 4.2
Database > OEEDB
Dataset > CUSTOMERS
SIPath > @

Processing SI-path CUSTOMER-NAME-KW of CUSTOMERS

KeyLength      =          6
BTree Height   =          2
# Tree Records =         36
Contiguous pairs =        35 (100.00 %)
# Leaf Records =         35
# Keys         =       2983

LenPrefix #Records  #Keys  Avg.Keys/Rec  Max Keys/Rec  Optimal Keys/Rec
    0         31    2544         82         83         84
    1          3    288         96         98         99
    4          1    117        117        117        249

CPU 0:00:00.6 Elapsed 0:00:02
    
```

```

Processing SI-path CUSTOMER-NAME of CUSTOMERS

KeyLength      =      17
BTree Height   =       3
# Tree Records =      38
Contiguous pairs =    37 (100.00 %)
# Leaf Records =      35
# Keys         =    1001

LenPrefix #Records #Keys Avg.Keys/Rec Max Keys/Rec Optimal Keys/Rec
      0      25    677         27         28          29
      1      10    290         29         29          30
                                         CPU 0:00:00.5 Elapsed 0:00:02

Processing SI-path ADDRESS1-CITY-KW of CUSTOMERS

KeyLength      =       6
BTree Height   =       2
# Tree Records =      48
Contiguous pairs =    47 (100.00 %)
# Leaf Records =      47
# Keys         =    4317

LenPrefix #Records #Keys Avg.Keys/Rec Max Keys/Rec Optimal Keys/Rec
      0      41   3367         82         83          84
      1       2    190         95         98          99
      4       4    714        179        248         249
                                         CPU 0:00:00.6 Elapsed 0:00:01
Total time :                               CPU 0:00:03.7 Elapsed 0:00:16
    
```

CHAPTER 7:

SITRACE utility

SITRACE is used to trace all IMAGE intrinsics and SUPERDEX intrinsics called in a program. It is very useful for debugging or logging all data base updates with SUPERDEX.

Activating SITRACE

To active the trace, the JCW SITRACE has to be set to a non-zero value before running the program to be traced:

```
:SETJCW SITRACE = n
```

Where n =	1	External IMAGE calls trace
>=	2	Plus, return status trace
>=	3	Plus, full dump of lock descriptor
>=	111	Internal SUPERDEX calls trace

Function

When processing DBOPEN, SUPERDEX checks if the **SITRACE** JCW is set to a non-zero value. If so, an internal flag is set to generate trace information.

❖ **The JCW is originally checked by DBOPEN, so it must be set before opening the data base. If the JCW is 0 at DBOPEN, the trace is disabled and all other procedures will skip the trace. If the JCW is not 0 at DBOPEN, the trace is enabled and all procedures recheck the current value of the JCW. This allows the user to "BREAK" a running program and RESUME it after changing the JCW, which will change or even stop the trace. Because each procedure rechecks the JCW, the trace facility will slow the process by about 5%.**

Redirection of Output

Trace output is sent to the file SITRACEF, which is defaulted to \$STDLIST and opened with Access Options "share" and "append". This means the user may:

- a) trace access via several access paths (multiple DBOPENs to one or more databases)
and
- b) append more trace information to an existing file.

Output can be redirected by specifying a file equation for SITRACEF.

Examples:

A. Redirect output to another free terminal:

```
:FILE SITRACEF;DEV=nn
```

B. Append output to an existing trace file:

```
:FILE SITRACEF=oldtrace,OLD
```

C. Create a new file and redirect output to it:

```
:BUILD mytrace;REC=-80,3,F,ASCII;DISC=1000  
:FILE SITRACEF=mytrace,OLD
```

D. Specify file to be created by SUPERDEX:

```
:FILE SITRACEF=mytrace,NEW;REC=-80,3,F,ASCII;DISC=1000;SAVE
```

E. Redirecting output to a temporary file from a screen driven (such as VIEW screen) program:

```
:RUN ENQUIRE.SUPERDEX.SYS > ENQTRACE
```

❖ Examples A, B, and C allow a trace access via multiple access paths (multiple DBOPENS to one or more databases), whereas D only allows a trace access via the access path of the first DBOPEN.

❖ Since QUERY first issues a DBOPEN with an empty BASE parameter and *mode 0* (to get TurboIMAGE version information) before prompting, example D can not be used (only the first DBOPEN will be traced, and nothing else).

CHAPTER 8:

SIDRIVER utility

SIDRIVER is a utility which permits IMAGE intrinsics to be “driven” interactively. It executes an intrinsic and returns the elapsed and CPU times.

The commands are very similar to HP's DBDRIVER utility. There have been some modifications and additions to the standard DBDRIVER utility to support all advanced SUPERDEX retrieval capabilities, new intrinsic parameters and modes, and new intrinsics. To execute:

```
:RUN SIDRIVER.SUPERDEX.SYS
```

❖ Only the differences between SIDRIVER and DBDRIVER are documented.

The modified commands are:

!B new format is *#{;base}*, where:
 # is a number 1 to 5 (used to assign multiple databases)
 {;base} is the base name to open (only used during open)

!Q Always upshifted
!L Always upshifted
!I Always upshifted

The new commands are:

!P Password parameter, in lower case if necessary

/DION Turns the DISPLAY COMMANDS option on. Useful with \$STDIN= option
/DIOF Turns DISPLAY COMMANDS off
/STON Set a switch to always print the status array after a call
/STOF Turns status print off
/OCON Report octal status, in addition to decimal
/OCOF Turns octal status print off
/HXON Report hex status, in addition to decimal
/HXOF Turns hex status print off

New intrinsics are callable by the following commands:

DE DBERASE
DX DBDELIX
PX DBPUTIX

This version of SIDRIVER does not support the **PRIV** entry point.

SECTION 7:

SuperSELECT

SuperSELECT was created to provide a simple means of changing a serial read to a SUPERDEX read with no program changes to existing software. This is especially helpful when using third-party software, or when source code is not available.

SuperSELECT works by interrupting a serial read being executed and replacing it with a SUPERDEX read. It can be executed using one of four methods.

The only restriction is that the multi-database relational access is not available. This is because SuperSELECT does not open or process any data. It sets some switches and builds a temporary file of the selection information. When the user program executes, the serial read will be intercepted and the data stored in the temporary file will be used to replace the serial read with a SUPERDEX read.

❖ **Within any of the four methods of execution, it is important to remember that only datasets and SI-paths that exist in the database that the user program serially reads are valid.**

Invoking SuperSELECT

There are no special requirements to invoke SuperSELECT. Simply execute SuperSELECT immediately prior to the normal execution of the user program.

SuperSELECT - Method 1

The first method for executing SuperSELECT is for testing and on-the-fly selections. SuperSELECT will loop through the prompts, allowing multiple arguments for a single SI-path, multiple SI-paths for a single dataset, and multiple datasets for a single database. This allows for full relational access within the database accessed by the user program.

Run the SuperSELECT program:

```
:RUN SUPERSEL.SUPERDEX.SYS
```

```
SuperSELECT Version 4.2
```

At this point SuperSELECT will loop through three prompts.

The first is the Dataset prompt:

```
Dataset>
```

Enter the name of the dataset that is located in the database opened by the user program to use against the selection. Press `RETURN` to exit SuperSELECT.

Next, the SIpath prompt will be displayed:

```
SIpath>
```

Enter the SI-path name to use against the selection. The SI-path must be one that is located in the dataset specified in the previous prompt. Pressing `RETURN` will cause SuperSELECT to return to the Dataset prompt.

Finally, the argument will be prompted for:

```
Argument>
```

Enter any valid SUPERDEX argument, including relational and Boolean operators. Once the argument has been entered the message "Record written", will be displayed for verification to the user.

SuperSELECT will then prompt for more arguments, until `RETURN` is pressed at the prompt. This allows for multiple search values to be entered. When `RETURN` is pressed, SuperSELECT will back up to prompt for another SI-path.

```
:RUN SUPERSEL.SUPERDEX.SYS  
  
SuperSELECT Version 4.2  
Dataset> CUSTOMERS  
SIpath> CUSTOMER-NAME-KW  
Argument>  
~UNI@ AND CHU@;  
Record written  
Argument>  
RETURN  
SIPath> RETURN  
Dataset> RETURN  
End of Program  
:RUN USERPROG
```


In the above example, SuperSELECT is used to change the serial search in USERPROG. When **USERPROG** executes, a SUPERDEX selection against the **CUSTOMER-NAME-KW** keyworded SI-path in the **CUSTOMERS** dataset will be done. The records selected will be the records that contain a word starting with **UNI** and a word starting with **CHU**.

If SuperSELECT is executed twice, before any serial read program, the temporary file will still exist. SuperSELECT will then display a message and ask if the older file should be purged.

```
Error Closing SuperSELECT file!
  Temporary file already exists.  Purge File (Y/N)?
```

If the older temporary file should be purged, enter **Y**, otherwise enter **N**.

SuperSELECT - Method 2

The second method for running SuperSELECT is used primarily prior to a batch run of a program, where there will only be one argument entered and that argument is known prior to when the batch job executes.

On the RUN command, the **;INFO** string is used to pass the dataset name, SI-path name, and the single argument value. For example:

```
:RUN SUPERSEL.SUPERDEX.SYS;INFO="CUSTOMERS;CUSTOMER-NAME-KW;~UNI@ AND CHU@"

SuperSELECT Version 4.2
Dataset>
CUSTOMERS
SIPath>
CUSTOMER-NAME-KW
Argument>
~UNI@ AND CHU@
Record written
```

This run command will set the same selection criteria as the example for method 1. The format of the **;INFO** string is:

```
:RUN SUPERSEL.SUPERDEX.SYS;INFO="dataset;si-path;argument;"
```

The rules for **dataset**, **si-path**, and **argument** are the same as described in the introduction of SuperSELECT.

If the SuperSELECT temporary file still exists, SuperSELECT will then display a message and ask if the older file should be purged.

```
Error Closing SuperSELECT file!
  Temporary file already exists.  Purge File (Y/N)?
```

If the older temporary file should be purged, enter **Y**, otherwise enter **N**.

SuperSELECT - Method 3

Method 3 of SuperSELECT is used primarily for a process that will accept the argument values from the user, and then a batch job is run, using the values. This allows a simple program to be written that will prompt the user for the arguments and then write the selection criteria to a flat file. The batch job will always run SuperSELECT, specifying that the flat file should be used.

This is done by running SuperSELECT like:

```
:RUN SUPERSEL.SUPERDEX.SYS;INFO="^ARGFILE"
```

This tells SuperSELECT to read the dataset, SI-path, and argument from the file *ARGFILE*. The file name can be fully qualified, including the group and account. The format of the **;INFO** string is:

```
:RUN SUPERSEL.SUPERDEX.SYS;INFO="^filename.group.account"
```

The file layout is similar to the **;INFO** string format used in method 2. It should be an unnumbered flat file, that includes the dataset name, si-path name, and the argument, separated and terminated by semicolons, and no blank lines:

```
dataset;siopath1;argument1;
dataset;siopath2;argument2;
.
.
.
```

For our example, the *ARGFILE* would contain one line:

```
CUSTOMERS;CUSTOMER-NAME-KW;~UNI@ AND CHU@;
```

When SuperSELECT runs it will display the values as it processes:

```

:RUN SUPERSEL.SUPERDEX.SYS;INFO="^ARGFILE"
SuperSELECT Version 4.2
Dataset>
CUSTOMERS
SIPath>
CUSTOMER-NAME-KW
Argument>
~UNI@ AND CHU@;
Record written
    
```

If the SuperSELECT temporary file still exists, SuperSELECT will then display a message and ask if the older file should be purged.

```

Error Closing SuperSELECT file!
  Temporary file already exists.  Purge File (Y/N)?
    
```

If the older temporary file should be purged, enter **Y**, otherwise enter **N**.

SuperSELECT - Method 4

Method 4 is a very powerful and user-friendly way to execute SuperSELECT on-line. It is used to allow a user to run SuperSELECT, input the argument(s), and automatically run the user's program, all through a data-entry screen.

This allows for great flexibility and automation for SuperSELECT, without having to train the user on how SuperSELECT works. Screens can be customized to match the appearance that the user expects, which will allow SuperSELECT to become an integral part of the user's application.

Before SuperSELECT is executed, enter a file equation for **SSSCREEN**, which can be fully qualified. Then run SuperSELECT without the **;INFO** string. SuperSELECT will open and process the customized screen.

```

:FILE SSSCREEN=CUSTOM.group.account
:RUN SUPERSEL.SUPERDEX.SYS
    
```

The screen file is a file built and designed by the programmer. Each line simply consists of a two (2) character command code, followed by a 78 character command or comment:

```

CCXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  ^-Command or Comment
  ^-Command Code
    
```

The valid command codes are:

Code	Command	Description
<i>Spaces</i>	Any comment or blank	Does nothing, only for internal comments.
AA	DATASET ; SIPATH ;	ACCEPT-ARGUMENT: Accepts any valid SuperSELECT argument from the user, and writes out the data.
D	Any string, including escape sequences	DISPLAY: Displays the Command portion without a carriage return.
DC	Any string, including escape sequences	DISPLAY-CARRIAGE RETURN: Displays the Command portion and then executes a carriage return.
DP	Any string, including escape sequences	DISPLAY-PROMPT: Displays the Command portion, then waits for the user to press RETURN . No data is accepted.
R	Any valid program name	RUN: Executes the program specified as a son process. No RUN parameters are valid.
S	DATASET ; SIPATH ; ARGUMENT ;	SET: Sets a fixed argument value for the dataset and SI-path
W	DATABASE[; PASSWORD] [; DATASET[; SIPATH]]	WINDOW: Sets up the WINGSPAN windows interface for SuperSELECT. The database name is required. The password, dataset, and SI-path are optional. If the password is not in the record, the user will be prompted for the password. If the dataset is not in the record, the user will be able to choose a dataset. If the SI-path is not in the record, the user will be able to choose an SI-path from the dataset.

Following is an example (the Command Codes are in bold for documentation purpose):

```

This is an example of a SuperSELECT Screen file.
All of the commands codes except for Window are used in this example. Of
course, they can be used in any order, and they are not
case sensitive.
DC(escape home and clear)
DC                               SuperSELECT Example
DC
DC  This is an example of how SuperSELECT screen files should be
DC  built. This example will prompt for a part description
DC  keyword for dataset ORDER-LINES and SI-path PART-DESC.
DC  It will also force an argument of "S" (shipped) for the
DC  SI-path SHIPPED-FLAG.
DC
DC          First, is the prompt for PART-DESC.
DC
DC  [notice we do not carriage return on the following line]
D  Please enter the Part Description argument:
AAORDER-LINES;PART-DESC;
DC
DC          Next, we force the shipped status.
S ORDER-LINES;SHIPPED-FLAG;~AND S;
DC
DP  Press RETURN to continue:
DC  Now start the user's program
R  USERPROG

```

Now, the screen display of how to execute SuperSELECT with method 4 and what is displayed:

```

:FILE SSSCREEN=USERSCRN
:RUN SUPERSEL.SUPERDEX.SYS
(screen is homed and cleared)
                               SuperSELECT Example

This is an example of how SuperSELECT screen files should be
built. This example will prompt for a part description
keyword for dataset ORDER-LINES and SI-path PART-DESC.
It will also force an argument of "S" (shipped) for the
SI-path SHIPPED-FLAG.

          First, is the prompt for PART-DESC.

Please enter the Part Description argument: ~RED AND BEAR@;

Press RETURN to continue: RETURN

```

At this point the user's program (USERPROG) from the **R** command would be executed. The program may display the report on the screen, or actually write out a print file.

SuperSELECT - Windows

The **(W)**indows command is available in the screen file method and is very powerful for end-user selections. This command allows the end-user many options in dataset selection, SI-path selection and argument selection. The format for the command is shown on page 7-6.

This first command only defines the database name which can be fully qualified. The user will be prompted for the database password. After a valid password is entered, the user can select which dataset(s) to use, which SI-path(s) to use, and the arguments.

```
W OEDB;
```

The next command contains the database password in the command. Since the screen file is not a privileged file, it is not recommended to keep the password in this file.

```
W OEDB.DEMO.SUPERDEX;CLERK;
```

The next command does not contain the password, but does contain a dataset name. This means the user must know the database password, but they will not be able to select a dataset. Since the dataset is defined, the dataset choice window will not be displayed.

```
W OEDB.DEMO;;CUSTOMERS;
```

If the SI-path should be forced, the SI-path name should be added to command, as follows. This will require the user to know the password, but will not allow them to change datasets or even the path to search by.

```
W OEDB;;CUSTOMERS;CUSTOMER-NAME-KW;
```

If the command file only has the database, as in the first example, the process will first ask for the database password.

```
RUN SUPERSEL.SUPERDEX.SYS  
OEDB Password>
```


The up and down arrows are used again to highlight and select a SI-path to use in the search. If the SI-path name was passed in the command, this screen will not be displayed and the SI-path name will already be displayed at the Path: prompt.

```
==      S U P E R S E L E C T      ==  
==      (C) Bradmark 1992      ==  
  
|-----|  
|Base: OEDB|  
|Set: CUSTOMERS|  
|Path: CUSTOMER-NAME-KW|  
|-----|  
  
  
  
  
  
  
  
  
  
  
_ Selection.. _____  
|XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX|  
|-----|
```

APPENDIX A:

**TurboIMAGE/XL Interface
to Third Party Indexing Products
SPECIFICATIONS**

INTRODUCTION

Appendix A is a copy of the document provided by Hewlett-Packard to Bradmark Technologies as the specifications for the Third Party Indexing Interface (TPI).

Because of differences in terms and new terms, we have marked certain terms with *italics*. These terms are defined, along with other comments, to help clarify the interface. Additions to the specifications by Bradmark are set of with [] and bold-faced.

Additionally, because of the design of SUPERDEX there is only one index or key type. This means SUPERDEX is much easier to implement. We recommend that these specifications be disregarded **IF YOU ARE ONLY INTERFACING WITH SUPERDEX**. If the indexing product is unknown, we recommend following these specifications.

Finally, this document is written to introduce the "new" capabilities within the TurboIMAGE/XL intrinsics. For current users of SUPERDEX, you will find these capabilities are not actually "new" as you have already had them for many years.

August 1, 1992

This document describes the standard (The use of the word 'standard' does not indicate a standard beyond the scope of TurboIMAGE/XL. It is not comparable to the SQL or ANSI standards. Instead, standard refers to the minimum set of features in TurboIMAGE/XL that all third party vendors agree to provide.) interface between Hewlett-Packard's TurboIMAGE/XL and third part indexing products. It provides an overview of the interface, discusses how the interface works, and details the enhancements to each of the TurboIMAGE/XL intrinsics.

OVERVIEW

Hewlett-Packard will provide a generic interface for third party indexing products. This is being done so new methods of access -- *generic key searches, sorted sequential retrievals, keyword searches, and single or multiple key searches* may be provided.

These new features will be available through TurboIMAGE/XL and third party indexing products at an intrinsic level. The retrieval and update specifications will be created with Hewlett-Packard so new intrinsic modes, status conditions, and argument tokens may be defined within the TurboIMAGE/XL intrinsics for indexing use. When these new modes or argument tokens are used, TurboIMAGE/XL will transparently interact with the third party indexing product to provide the corresponding features. This common functionality defined with Hewlett-Packard will allow applications to be written that take advantage of the third party indexes without the advance knowledge of a specific third party product installed on the system.

The third party indexing products may provide features beyond those that are standardized. For this purpose, specific ranges of modes, status conditions, and argument tokens within the TurboIMAGE/XL intrinsics have been set aside. Each third party vendor may use these at its own discretion, and for its individual purposes.

Additionally, the third party vendors may include additional intrinsics that provide features not directly translated into the TurboIMAGE/XL intrinsics. These intrinsics are entirely the responsibility of the third party vendor.

HOW THE INTERFACE WORKS

The interface between TurboIMAGE/XL and third party indexing products is fundamentally an intrinsic interface. The TurboIMAGE/XL intrinsics will be modified so that they update and access the third party indexes. They will do this by calling internal and restricted routines provided by the third party vendor. These third party routines will update and access their indexes, and then return to the TurboIMAGE/XL intrinsic.

In most cases, the flow of the enhanced TurboIMAGE/XL intrinsic will be as follows:

- The TurboIMAGE/XL intrinsic performs its standard TurboIMAGE/XL function.
- The TurboIMAGE/XL intrinsic determines if a call to the third party routine is appropriate, and if so, calls the routine.
- The third party routine acts appropriately against its indexes, and then returns to the calling TurboIMAGE/XL intrinsic.
- If the third party routine indicates that it completed successfully, TurboIMAGE/XL completes the intrinsic by performing any logging and recovery steps. Unsuccessful intrinsics are backed out by the MPE XL Transaction Manager or TurboIMAGE/XL's dynamic rollback.
- The TurboIMAGE/XL intrinsic returns to the calling program.

Using this method, the TurboIMAGE/XL intrinsics will insure that changes to the third party indexes are synchronized with changes to the TurboIMAGE/XL database, including transaction logging, dynamic rollback, and other recovery strategies. This provides that no programming changes will be necessary during update operations.

Also using this method, the TurboIMAGE/XL intrinsics will take advantage of the retrieval capabilities of the third party indexes. The intrinsics DBFIND and DBGET will continue to provide the traditional methods of access -- forward and backward serial reads, forward and backward chained reads, hashed, and directed reads. Additionally, DBFIND and DBGET will provide new methods of access -- *generic key searches, sorted sequential retrievals, keyword searches, and single or multiple key searches.*

INSTALLING THE INTERFACE

The vendors of third party indexing products will provide utilities and support for installing their products. The installation process will contain at least three important steps, all performed by the third party installation procedures.

The first installation step is to move the third party executable library and utilities into the PUB group of the SYS account. The executable library will contain the software necessary to implement this interface, as well as other procedures at the discretion of the third party vendor. The third party index configuration utility may be used to verify proper installation of the third party executable library. **[The SUPERDEX library is XLSUPRDX.PUB.SYS and the utilities are located in the SUPERDEX group of the SYS account, not the PUB group.]**

After the first installation step, TurboIMAGE/XL will be able to call the third party executable library whenever a TurboIMAGE/XL intrinsic is executed. The third party procedures will provide the extended TurboIMAGE/XL features described in this document. They may also provide other features at the discretion of the third party vendors. Range of modes, status conditions, and argument tokens have been reserved for indexing purposes.

The second installation step is configuring the third party indexes for a database using a utility furnished by the third party. These configured indexes will reside either in data sets within the configured database or external files. If the indexes reside in external files they will follow the naming convention of "dbname0A, dbname0B, ..., dbname0Z, dbname1A, ..., dbname9Z." After installation, TurboIMAGE/XL will consider both the TurboIMAGE/XL data and the third party indexes as a single logical database. This logical database may be stored and restored and managed with DBUTIL. **[Based on discussion with the Hewlett-Packard TurboIMAGE laboratory, SUPERDEX continues to utilize one to eight data sets within the database.]**

The index files will not be apparent to the user. Neither schema files for DBSCHEMA, nor the listing from a QUERY FORM command will show the third party index files. Traditional DBINFO calls will not include the index files in their list of sets in the database, and other TurboIMAGE/XL calls, such as DBPUT, DBDELETE, DBUPDATE, DBFIND and DBGET, will not be allowed against these files.

The third installation step is updating the TurboIMAGE/XL root file to show the presence of the indexing product on that database. This can be done through DBUTIL's "ENABLE" command or may be done through a third party utility when configuring for indexes in the second step. Until the root file shows indexing information, the database updates will not trigger updates to third party indexes. **[DBUTIL's "ENABLE" will not work for turning SUPERDEX on. One of SUPERDEX's utility programs, such as SUPERDEX.SUPERDEX.SYS or SIMAINT.SUPERDEX.SYS, must be used.]**

There can be multiple third party products installed on the system. However, only one third party product can be configured for a database at a time. If it is desired to configure a database for the other third party product on the system, the database must be disabled for indexing first and then configured for the other product.

OPENING THE DATABASE

When a database is opened using DBOPEN, TurboIMAGE/XL will check for the presence and identity of third party indexing. This will be known by interrogating the TurboIMAGE/XL root file modified in the installation process. If third party indexing is configured, DBOPEN will call the related third party routine which will prepare for using the indexes.

Once third party indexing is enabled for a database, third party indexing will automatically be activated from TurboIMAGE/XL intrinsics. However, it is possible in some applications that the new wildcards honored by third party indexing will conflict with existing data values for a particular application. In these cases, third party indexing enhancements to DBFIND and DBGET can be disabled for a specific application by adding 100 to the DBOPEN mode or by setting the IMAGETPI JCW to 100. Verification of the IMAGETPI JCW and appropriate processing will be done by the third party product.

UPDATE THE DATABASE

Once the third party product is installed and a database has been enabled for third party indexing, third party indexes will be appropriately updated whenever a TurboIMAGE/XL intrinsic is called. No other changes are needed by the system manager, the programmer or the user.

RETRIEVING DATA FROM THE DATABASE

The third party indexes provide two methods of retrieving data that are fundamentally different from TurboIMAGE/XL. These methods supplement, but do not replace, the existing methods of retrieval in TurboIMAGE/XL.

The first method, *generic key search*, allows wildcard values to be entered as part of a key. In response, all records meeting that wildcard value will be available. The records will be displayed in sorted order by key value, eliminating the need for sorting in many situations. Also, this method allows *composite keys* [***concatenated indexes***] which are comprised of components from several fields

in a data set, and these components are not required to begin and end on field boundaries. [**generic key search in SUPERDEX is called *Indexed Access*. It simply means looking at a single B-tree with a start and stop location and returning the records in sorted sequential order by index.**]

The second method, *keyword searches* and *single or multiple key searches*, allows searches against the individual words or numbers within fields. Searches can also be combined across multiple fields as well as sets which is called *multiple key searches*. This allows searches containing complex criteria to be performed with complete indexed access to the database. When a keyword search defines a set of data entries, the set of data entries may be refined with subsequent keyword searches. A DBFIND mode is available to cancel the last keyword search refinement. [**SUPERDEX separates *keyword searches* and *single or multiple key searches*. Because of the design of other third party indexing products, keywording requires a special type of index. Keywording in SUPERDEX is only an option on an index. *Single or multiple key searches* is called *Relational Access* in SUPERDEX. Throughout this document, whenever terms such as *keywording*, *keyword searches*, and *single or multiple key searches* are used, *Relational Access* should be implied. There is no special programming necessary in SUPERDEX to support true keywording, which is "... searches against the individual words or numbers within fields."**]

In both of these methods, DBFIND is used to qualify a "*working set*" [***SI-Chain* in SUPERDEX**] of records. With *generic keyed retrieval* [***Indexed Access***], the working set is the specified range of records. For example, the argument ">=JONES<=SMITH" produces the working set of JONES through SMITH. Similarly, the argument ">=SMITH" produces the working set of SMITH through the end of the file. With *keyword retrieval* or *single or multiple key retrieval* [***Relational Access***], the working set [***Virtual SI-Chain***] begins with the first DBFIND, and can then be refined with successive calls to DBFIND.

DBGET is used to retrieve the entries of the working set qualified by DBFIND. A current record pointer is maintained to read forward and backward within the working set of records. DBGET will return a status 14, or status 15, when a boundary of the working set is reached. Additional modes of DBGET are available to continue reading beyond the bounds of the working set and these modes return a status of 10 or 11 when the beginning or end of the file is encountered.

ENHANCING EXISTING RETRIEVALS

Predominantly, new types of third party indexing retrieval will require using new modes and programming constructs. As such, they will generally require programming changes in existing programs. However, in one case, it will be possible to enhance existing retrievals without any changes to these programs.

Enhancing existing retrievals can occur when reading a TurboIMAGE/XL chain in a detail data set of data type U or X. In this situation, it will be possible to enter wildcards (@, #, ?, defined later), and retrieve all records that meet the indexing criteria. DBFIND mode 1 will return a chain count which equals the sum of all the qualified entries matching the wildcard value. DBGET modes 5 or 6 will then process through the qualified entries as though they were one chain.

[**Again, because of SUPERDEX's design, there are additional capabilities when a data set has a chain. *Relational Access* is available through DBFIND mode 1/DBGET mode 5. Additionally, SUPERDEX does not require a special mode for binary items, although the new modes can be used.**]

For data types other than U or X, DBFIND and DBGET use different modes. Tokens cannot be allowed in a standard DBFIND mode 1 for binary items. Since binary items expect binary arguments, an ASCII token cannot be properly interpreted. For example, the character ">" followed by the integer 100, representing the argument "anything greater than 100", is synonymous with the full integer value 15,972 when both are viewed as binary arguments.

Application developers who are programming for all indexing packages should use the DBFIND/DBGET modes specifically allocated for *generic key* [***Index Access***] and *keyword retrievals* [***Relational Access***] to maintain portability between packages.

NEW RETRIEVAL CONSTRUCTS

Each method of access in TurboIMAGE/XL requires a different programming construct. To retrieve detail records based on a TurboIMAGE/XL search item, a DBFIND is called against the detail search item, followed by several calls to DBGET as the chain is read. Alternatively, for serial reads, a DBCLOSE is called to rewind the set, and then DBGET is called to read through the data set. A calculated read in a master set requires a single DBGET call, however, in this situation, an argument is supplied to DBGET. The modes, arguments, and organization of the intrinsics are different in each of these situations.

Generic key searches, sorted sequential retrievals, keyword searches, and single or multiple key searches, will also require different modes, arguments, and programming constructs. *Generic key search and sorted sequential retrieval* will allow wildcard tokens and relational operators in the argument of DBFIND. *Keyword retrieval and single or multiple keyword/key searches* will involve successive calls to DBFIND using both Boolean and relational operators in the argument. All of these calls to DBFIND will be allowed on both master and detail data sets, and these searches will be followed by calls to DBGET using new modes.

The basic programming construct for each of the methods of access methods are shown on the next page. The specific modes and argument tokens are described in detail in the definitions of the TurboIMAGE/XL intrinsics in the following section.

[If only programming for SUPERDEX, valid retrieval modes and arguments are the same, although the programming construct may change. The general program construct should be to process DBFIND mode 1, with any argument the user enters, until the user has completed entering arguments (could be only one DBFIND, or many DBFINDs), followed by DBGET mode 5 until a condition word of 15 (end-of-chain) is returned. There are variations that can be used, but this simple programming construct handles most SUPERDEX retrievals.]

DEVELOPING THIRD PARTY APPLICATIONS

Some of the basic features provided by the third party indexing products have been standardized in this interface. This allows the developers of third party applications to take advantage of these features without the advance knowledge of a specified third party product installed on the system. **[Useful for software development companies who want to support any third party indexing product, or for very large companies who allow remote sites to purchase any third party indexing product.]**

To facilitate the development of these applications, new modes of DBINFO are provided which determine if a third party indexing product is installed, and describe the installation of *keys* [**Third party indexes**]. By using these new modes, developers can fully customize their programs to use third party indexes.

Retrieval Contracts for TurboIMAGE


Calculated Read

DBGET mode 7


Direct Read

DBGET mode 4


Serial Read

DBCLOSE mode 3
 DBGET mode 2 or 3
until EOF or BOF


Chained Read

DBFIND mode 1
 DBGET mode 5 or 6
until EOC or BOC

Generic Key Search

DBFIND mode 1 using
wildcards or mode 11
 DBGET mode 5 or 6
until EOC or BOC

Single or Multiple Keyword/Key Search

DBFIND mode 12 for
each keyword/key field
 DBGET mode 25 or 26
until EOC or BOC

TurboIMAGE/XL UTILITIES

DBUTIL

DBUTIL will provide the following support to the third party indexing products:

- **DISABLE** The DISABLE command will include the option:

DISABLE dbname FOR INDEXING

This option will disable the database for third party indexing. After a database is disabled, any updates to the database will not be reflected in the third party indexes. Hence, it is recommended that indexes be re-validated or reconfigured after disabling a database for indexing. Using this command requires exclusive access to the database.

- **ENABLE** The ENABLE command will include the option:

ENABLE dbname for INDEXING

This option will enable the database for third party indexing. A database can also be enabled for indexing by a third party configuration utility when configuring a database for indexing. If a database is disabled for indexing, subsequent updates to the database will not be reflected in the third party index files. Hence, it is recommended that indexes be reconfigured before enabling the database for indexing. This command requires exclusive access to the database.

[SUPERDEX will not allow this command to enable a database. Because of possible database structural modifications, SIMAINT or SUPERDEX must be used to enable a database for SUPERDEX indexing.]

- **PURGE** The PURGE command will purge the logical database, consisting of both the database and third party index files, if any, regardless of Indexing being ON or OFF. In other words, if there are any existing extraneous third party index files corresponding to the database name being purged, they will be purged as well. For example, if a database 'ORDER' was configured for indexing, but later disabled for indexing, and is being purged, all existing third party index files starting with ORDER0A will be purge as well.

[This works the same for ANY SUPERDEX version.]

- **RELEASE** The RELEASE command will release the security for the logical database, consisting of both the database and third party index files, if any.

[This works the same for ANY SUPERDEX version.]

- SECURE The SECURE command will secure the logical database, consisting of both the database and third party index files, if any.

[This works the same for ANY SUPERDEX version.]

- SHOW The SHOW command will additionally display the status of third party indexing for the database.

DBSTORE/DBRESTOR

DBSTORE/DBRESTOR or its successor will store and restore the logical database, consisting of both the database and third party index files, if any. DBSTORE/DBRESTOR support is not targeted for first release of this interface. **[This will continue to work as always for ANY SUPERDEX version.]**

DBCHANGE PLUS

DBCHANGE PLUS will copy the logical database, consisting of both the database and third party index files, if any. DBChange Plus will also notify third party indexing products of changes to the structure of a database. **[Any database utility program that copies a database will continue to copy all of SUPERDEX as always. The DBChange Plus notification version will not support SUPERDEX in the first release of this interface. DBGENERAL continues to work as always.]**

INTRINSIC DEFINITIONS

Naming Convention for Additional Intrinsic

This interface provides that third party vendors may include additional intrinsics that provide features not directly translated into TurboIMAGE/XL intrinsics. These intrinsics will begin with the prefix TPI in order to distinguish them as belonging to the third party vendors, and to avoid naming conflicts in the future. Exceptions to this will be approved by Hewlett-Packard. It is expected that each third party vendor will request that the intrinsics they currently distribute will be allowed and reserved. It is expected that Hewlett-Packard will approve the inclusion of these intrinsics and that Hewlett-Packard will not use these names in the future. **[The SUPERDEX intrinsics DBDELIX and DBPUTIX are located in the SUPERDEX XL and are approved by Hewlett-Packard.]**

New DBTPIINFO Intrinsic

A new intrinsic DBTPIINFO will be supported by Hewlett-Packard and the third party vendors. This intrinsic will supply information about the third party indexing products installed on the system as well as the external and internal index files created for a database, without the need for the caller to open the database. The following is the intrinsic definition:

DBTPIINFO (Base, Qualifier, Mode, Status, Buffer)

base parameter: Same as current TurboIMAGE/XL
Ignored for mode 1. Mode 2 requires database name.

qualifier parameter: See specific information under each mode

mode and buffer parameters (buffer parameters in half words):

- 1 Returns information on third party indexing products installed and available for third party indexing.

qualifier: ignored

buffer layout:

1	Number of Products Installed. Zero if none installed.
25(n-1) + 2	Product name.
25(n-1) + 22	Version number.

- 2 Identifies third party index files configured for a database.

qualifier: ignored

buffer layout:

1	Number of external files. Zero, if none. [Always 0 in SUPERDEX]
2	Number of internal files. Zero, if none. [1 - 8 in SUPERDEX]
3-10	Reserved for future use. Blanks for now.

Status parameter: Same as current TurboIMAGE/XL status array.

The status array elements will have the following values:

- | | |
|-------|--|
| 1 | Zero when successful, even when no TPI product is installed on the system for mode 1, or when the database is not enabled for indexing for mode 2. In these cases, interrogate the appropriate elements of the buffer parameter for a value of zero. Non-zero status values of 3nnn and -3nnn are reserved for non-standard errors from DBTPIINFO. |
| 2 | Length of the buffer in half-words.
For mode 1, common values are 26 and 51, however, it will be 1 when no third party product is installed on the system. For mode 2, it will always be 10. |
| 3,4,5 | Will be zeroes. |
| 6 | Pcode of DBTPIINFO (480). |
| 7,8 | For mode 2, address of base parameter. |
| 9 | Mode. |
| 10 | Zero. |

DBFIND

DBFIND (base, dset, mode, status, item, argument)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

- 1 If no special tokens are specified or third party indexing is not configured for this item, and item is a standard TurboIMAGE/XL search item on a detail data set, then perform the equivalent of a standard DBFIND. Otherwise, set the record pointer based on *generic key* [***Index Access***], returning "chain count" as described below. Other vendor specific options are allowed using special characters in the argument, but these are not part of the standard. **[This mode, based on the standard, only supports single value (partial, generic, and range) retrievals for character type indexes (X or U). SUPERDEX allows retrievals on non-character type indexes, along with full *Relational Access* (Boolean operators) with DBFIND mode 1 (See SUPERDEX User Manual, Section 5.)]**

Special note on binary items, and third party keys containing binary components:

Tokens and wildcards are not supported in the argument parameter of binary items. Since binary items expect binary arguments, a token cannot be properly interpreted. For example, the character ">" followed by the integer 100, representing the argument "anything greater than 100", is synonymous with the full integer value 15,972 when both are viewed as binary arguments. DBFIND mode 11 should be used to specify ranges on binary items. **[SUPERDEX allows DBFIND mode 11, but does not require it. For binary values, the argument can be entered as an ASCII value. For example, the argument could be ">=100 " (all ASCII). SUPERDEX will convert the 100 to the correct binary value.]**

- 10 Performs a standard TurboIMAGE/XL DBFIND on a TurboIMAGE/XL detail search item. DBFIND will be used to qualify a working set of records without the assistance of third party indexing functions. **[This would only be necessary if the IMAGE item name is also a SUPERDEX SI-Path name. If the IMAGE item name is not a SUPERDEX SI-Path name, only IMAGE will be used in mode 1.]**

- 11 Provides range retrieval for binary data items (I, J, K, P, Z, R, E). For the reason stated above, ranges are not allowed for binary items using mode 1. With mode 11, the argument buffer will contain both the start and stop values. The size and data type of each sub-argument is the exact size and data type of the item. **[As stated earlier, SUPERDEX will convert the ASCII value from DBFIND mode 1 to binary. The program does not need to.]**

To specify open-ended ranges, either the start or stop value must contain the minimum or maximum value for the index. For example, to specify the argument ">=100", the first part of the argument buffer would contain a binary 100. The second part of the argument would contain the highest possible value for the index.

- 12 Performs a *keyword search* qualification, and returns "chain count". The chain count is the number of records qualified so far based on this retrieval. **[As stated earlier, because of SUPERDEX's design there is no reason to necessitate a special mode for "keyword search". "Keywording" is simply an option, and index qualification can be made WITH NO SPECIAL ARGUMENTS with DBFIND mode 1.]**

It is possible to perform *multiple key searches* [**Relational Access**]. To accomplish this, DBFIND's with mode 12 are called in succession. In these cases, the argument parameter also describes the relationship between the DBFIND calls. See the argument parameter section. **[Again, because of SUPERDEX's design DBFIND mode 1 can continue to be used to execute multiple key searches or Relational Access. DBFIND mode 12 simply enforces Relational Access during the DBFIND.]**

- 13 Restores back to the most recently qualified list for *single or multiple keyword/key retrievals*. **[What this says is, during a DBFIND mode 12 a "previous" qualified list will be created. For example, if the first DBFIND contains an argument of "A@", and the second contains an argument of "and B@" then the previous list will contain just the "A@" records. After the second DBFIND (with "and B@") a call to DBFIND using mode 13 will replace the qualified list to the "A@" records and the user can now enter "and C@". The "and C@" will apply to the complete list of "A@" records, not just the "A@ and B@" records.]**

- 21 Same as mode 1, but does not return a "chain count" **[Same as old SUPERDEX mode 10]**. The argument rules for mode 21 are identical to mode 1 **[All SUPERDEX extensions to mode 1 apply to mode 21]**.

- 22 Same as mode 11, but does not return a "chain count". The argument rules for mode 22 are identical to mode 11.

- 23 Same as mode 12, but does not return a "chain count". The argument rules for mode 23 are identical to mode 12.

Note that the following modes (1nn-5nn) represent programmatic methods of enforcing the relational operators. It is also possible to apply the relational operators in a DBFIND mode 1 by using their corresponding tokens directly in the argument. Modes 1nn-5nn will not return a chain count.

- 1nn Sets record pointer before the first record which matches **[equal-to (=)]** the first nn words of the argument and qualifies a working set which matches the first nn words of the argument. If a negative number, then evaluates the first nn bytes. If 100, then sets the record pointer to record with lowest key **[index]** value and qualifies all entries in the data set. If 199, then the full key value **[full index length, minus the SI-Pointer]** is evaluated, regardless of size. If a key value matching the argument is not found, a status condition of 17 is returned.

- 2nn Sets record pointer before the first record greater than (>) the argument, evaluating the first nn words and qualifies a working set of all records which have a value greater than the argument. If a negative number, then evaluates the first nn bytes. If 200, then sets the record pointer to record with lowest key **[index]** value and qualifies all entries in the data set. If 299, then the full key value **[full index length, minus the SI-Pointer]** is evaluated, regardless of size. If a key value matching the argument is not found, a status condition of 17 is returned.

- 3nn Sets record pointer before the first record greater than or equal to (>=) the argument by evaluating the first nn words and qualifies a working set of all records which have a value greater than or equal to the argument. If a negative number, then evaluates the first nn bytes. If 300, then sets the record pointer to record with lowest key **[index]** value and qualifies all entries in the data set. If 399, then the full key value **[full index length, minus the SI-Pointer]** is evaluated, regardless of size. If a key value matching the argument is not found, a status condition of 17 is returned.

- 4nn Sets record pointer after the first record less than (<) the argument by reading from the end of the file **[SUPERDEX does not read any of the file. It only qualifies based on indexes.]**, evaluating the first nn words and qualifies a working set of all records which are less than the argument. If a negative number, then evaluates the first nn bytes. If 400, then sets the record pointer to record with highest key **[index]** value and qualifies all entries in the data set. If 499, then the full key value **[full index length, minus the SI-Pointer]** is evaluated, regardless of size. If a key value matching the argument is not found, a status condition of 17 is returned.

5nn Sets record pointer after the first record less than or equal to (\leq) the argument by reading from the end of the file [**SUPERDEX does not read any of the file. It only qualifies based on indexes.**], evaluating the first nn words and qualifies a working set of all records which are less than the argument. If a negative number, then evaluates the first nn bytes. If 500, then sets the record pointer to record with highest key [*index*] value and qualifies all entries in the data set. If 599, then the full key value [**full index length, minus the SI-Pointer**] is evaluated, regardless of size. If a key value matching the argument is not found, a status condition of 17 is returned.

Modes Reserved for Third Party Indexing Functions [**Extension**]

1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions [**extensions**].

status parameter:

The "chain counts" for third party indexed retrievals are shown in the chain count words of the TurboIMAGE/XL status array. Additionally, the condition word would reflect the various exceptional and error conditions.

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. (This range was picked because it is an unused range within TurboIMAGE/XL.) The values may be assigned and controlled by the respective vendors.

Calling Errors

-11 Bad base parameter
-21 Bad dset parameter
-31 Bad mode
-52 Bad item
-258 Invalid argument for index.
-259 Invalid mode for index.
-260 No previous list of qualified data entries.

Exceptional Conditions

17 No entry found matching argument

Values Reserved for Third Party Indexing Functions [Extensions]

3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

item parameter:

The item parameter may now reference third party keys as well as TurboIMAGE/XL detail data set search items.

argument parameter:

The argument parameter may now contain *generic key values* and *single or multiple keyword/key search values* and operators in additions to TurboIMAGE/XL search items. Generic key values consist of either 1) single value, optionally with wildcards and a relational operator, or 2) two values with relational operators which establish a range. Single or multiple keyword search values consist of one or more values, connected with Boolean or range operators. The minimum standard for relational operators and Boolean operators are shown later in this section. Single or multiple keyword searches performed through mode 12 will assume algebraic notation of Boolean operations. The argument parameter for mode 12 is terminated with a semicolon. **[What this says, is many different operators can be included in the argument (@,#,?, AND, OR, NOT, TO, >, <, =, ;).]**

The data type of the argument parameter depends on the mode used. The table below shows the expected data type for each mode:

<u>Mode</u>	<u>Data Type</u>
1	ASCII for ASCII items; Binary without wildcards for Binary items.
10	ASCII for ASCII items; Binary for Binary items
11	Binary for Binary items
12	ASCII
13	Ignored
21	ASCII for ASCII items; Binary for Binary items
22	Binary for Binary items
23	ASCII
1nn	ASCII, Binary, or a combination of ASCII and Binary
2nn	ASCII, Binary, or a combination of ASCII and Binary
3nn	ASCII, Binary, or a combination of ASCII and Binary
4nn	ASCII, Binary, or a combination of ASCII and Binary
5nn	ASCII, Binary, or a combination of ASCII and Binary

The argument parameter may now reference special tokens used by third party indexing products. The paragraphs which follow describe the minimum standard for special tokens. A third party indexing product may also employ operators from the following set of characters to provide additional features: ~!\$%^&*~+~.,[]{}|.

The wildcard tokens (? , #, and @) provide methods of searching for data without knowing all of the characters of a key. The wildcard token ? represents any single alphanumeric character. The wildcard token # represents any single digit. The wildcard token @ represents any number of alphanumeric characters, including no characters. Within the standardized interface, the tokens ? and # may exist anywhere in a key except for some restrictions noted in the next paragraph, and the @ may appear at the end of a key **[SUPERDEX allows two @s in some cases]**. Third party vendors are obligated to provide this implementation, but not restricted to this implementation. Wildcard tokens are not supported for binary items.

More than one relational operator may be used in an argument to represent a range. For example, "5 to 10" could be represented as ">=05<=10". When a wildcard (@, #, ?) is combined with more than one relational operator the wildcard(s) must be at the end of the wildcard value. In other words, an argument of ">ABC@<XYZ@" is allowed but an argument of ">ABC@DEF<UVW@XYZ@" is not allowed. **[SUPERDEX does not support a single > or < in the argument for this first release. Only the extensions of >=, <=, and <> are supported.]**

The Boolean operators listed below serve two purposes. The first purpose is to establish the relationship between keywords in the argument. For example, the argument "SMITH OR JONES" qualifies all records containing either SMITH or JONES. The second purpose of the Boolean operators is to specify how successive calls to DBFIND are related. If the argument begins with a Boolean operator, then the results of this DBFIND will be applied against the previous DBFIND using the stated operator. For example, the argument "AND SMITH OR JONES" would first qualify "SMITH OR JONES", and then this result would be AND'd with the previous DBFIND. **[SUPERDEX allows the AND to be processed first by specifying the order of precedence with parentheses. For example, "(AND SMITH) OR JONES" would process AND SMITH with the previous DBFIND, followed by OR JONES.]**

Operators and tokens are evaluated in the order presented below:

- " " Double quotes surrounding a string indicate it should be searched for, even though it is a token. For example "and" will search for the word "and", and "VT@" will search for the string "VT@".
- () Parentheses surrounding a string of Boolean or relational operators explicitly define the order in which the operators are to be evaluated (mode 12 [**or mode 1 in SUPERDEX**]).
- @ Any number of characters or digits.
- ? One character or one digit
- # One digit
- > "GREATER THAN" relational operator [**Not supported in the first release**]
- >= "GREATER THAN OR EQUAL TO" relational operator
- < "LESS THAN" relational operator [**Not supported in the first release**]
- <= "LESS THAN OR EQUAL TO" relational operator
- [<> "**NOT EQUAL TO" relational operator**
- TO Inclusive range operator (mode 12 [**or mode 1 in SUPERDEX. "5 to 10" is the same as ">=5<=10".**])
- NOT "(AND) NOT" Boolean operator (mode 12 [**or mode 1 in SUPERDEX**])
- AND "AND" Boolean operator (mode 12 [**or mode 1 in SUPERDEX**])
- OR "OR" Boolean operator (mode 12 [**or mode 1 in SUPERDEX**])

Argument Tokens Reserved for Third Party Indexing Functions
[Extensions]

The following tokens may be used for third party indexing functions beyond the standard. Note that any tokens from this list which are enclosed in quotation marks will be treated as data: ~!\$%^&*~+~:,[]{}.

DBGET

DBGET (base, dset, mode, status, list, buffer, argument)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

- 1 Standard TurboIMAGE/XL mode: Re-read
- 2 Standard TurboIMAGE/XL mode: **[Forward]** Serial read
- 3 Standard TurboIMAGE/XL mode: Backward serial read
- 4 Standard TurboIMAGE/XL mode: Directed read
- 5 If following a standard DBFIND along a TurboIMAGE/XL search item, then performs a standard forward chained read. If following a *generic key search*, read next record in the chain using third party indexing product. At the end of the **[index]** chain, an EOC status **[the standard IMAGE condition word 15]** is return. **[With SUPERDEX, DBGET mode 5 can be used to read records qualified from any DBFIND mode.]**
- 6 If following a standard DBFIND along a TurboIMAGE/XL search item, then performs a standard backward chained read. If following a *generic key search*, read previous record in the chain using third party indexing product. At the beginning of the **[index]** chain, an **[the standard]** BOC status **[condition word 14]** is return. **[With SUPERDEX, DBGET mode 6 can be used to read records qualified from any DBFIND mode.]**
- 7 Standard TurboIMAGE/XL mode: Calculated read
- 8 Standard TurboIMAGE/XL mode: Primary calculated read
- 11 Resets pointer to the beginning of the list of qualified records for *generic key search*. **[Resets pointer for any SUPERDEX DBFIND mode.]**
- 12 Moves pointer forward "n" entries in the qualifying list of qualified records for a *generic key search*, without actually retrieving any records. "n" is a 32-bit integer sent in the argument. **[Moves pointer for any SUPERDEX DBFIND mode.]**
- 13 Moves pointer backward "n" entries in the qualifying list of qualified records for a *generic key search* without actually retrieving any records. "n" is a 32-bit integer sent in the argument parameter. **[Moves pointer for any SUPERDEX DBFIND mode.]**

- 15 Same as DBGET mode 5, but is not inhibited by an end of chain status. DBGET mode 15 will return a status 10 when the end of file is encountered. **[Reads through the indexes in sorted order until the end of indexes. Useful when replacing KSAM operations.]**
- 16 Same as DBGET mode 6, but is not inhibited by a beginning of chain status. DBGET mode 16 will return a status 11 when the beginning of file is encountered. **[Reads through the indexes in descending sorted order until the beginning of indexes. Useful when replacing KSAM operations.]**
- 21 Resets pointer to the beginning of the list of qualified records for *keyword/key retrieval*. **[Exactly the same as DBGET mode 11 in SUPERDEX, and resets pointer for any SUPERDEX DBFIND mode.]**
- 22 Moves pointer forward "n" entries in the list of qualified records for *keyword/key retrieval*, without actually retrieving any records where "n" is a 32-bit integer sent in the argument. **[Exactly the same as DBGET mode 12 in SUPERDEX, and moves pointer for any SUPERDEX DBFIND mode.]**
- 23 Moves pointer backward "n" entries in the list of qualified records for *keyword/key retrieval*, without actually retrieving any records where "n" is a 32-bit integer sent in the argument. **[Exactly the same as DBGET mode 13 in SUPERDEX, and moves pointer for any SUPERDEX DBFIND mode.]**
- 25 Retrieves next record buffer from those qualified by *keyword/key retrieval*. **[Exactly the same as DBGET mode 5 in SUPERDEX, retrieving records qualified from any SUPERDEX DBFIND mode.]**
- 26 Retrieves previous record buffer from those qualified by *keyword/key retrieval*. **[Exactly the same as DBGET mode 6 in SUPERDEX, retrieving records qualified from any SUPERDEX DBFIND mode.]**

Modes Reserved for Third Party Indexing Functions [Extensions]

- 1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products (this range was picked because it is an unused range within TurboIMAGE/XL). The values may be assigned and controlled by the respective vendors.

Calling Errors

- 11 Bad base parameter
- 21 Bad dset parameter
- 31 Bad mode
- 51 Bad list length
- 52 Bad list or item

Exceptional Conditions

- 10 Beginning of file
- 11 End of file
- 12 Directed beginning of file
- 13 Directed end of file
- 14 Beginning of chain
- 15 End of chain
- 17 No entry found matching argument
- 18 Broken chain
- 50 Buffer is too small
- 62 DBG full
- 63 Bad DBG

Values Reserved for Third Party Indexing Functions [Extensions]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

list parameter: Same as current TurboIMAGE/XL [**SUPERDEX has extensions; see SUPERDEX User Manual, Section 5**]

buffer parameter: Same as current TurboIMAGE/XL

argument parameter: Same as current TurboIMAGE/XL

DBOPEN

DBOPEN (base, password, mode, status)

base parameter: Same as current TurboIMAGE/XL

password parameter: Same as current TurboIMAGE/XL

mode parameter:

1-8 Standard TurboIMAGE/XL modes

+100 Adding 100 to existing modes opens the database but disables third party indexed retrievals. This option affects the intrinsics DBFIND and DBGET only.

Note: Setting the JCW IMAGETPI to 100 before running a program [**opening a database**] will add 100 to DBOPEN modes 1-8 if third party indexing is installed.

Modes Reserved for Third Party Indexing Functions [Extensions]

1000/1799

The ranges of modes 1000 to 1799 and -1799 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

- 11 Bad base parameter
- 21 Bad dset parameter
- 31 Bad mode
- 32 Unobtainable mode
- 88 Database bad: Third party indexing was in process (index again).
- 89 Database bad: Restructuring was in process (restore database).
- 90 Root file bad: Unrecognized state: % octal integer
- 91 Bad root modification level
- 92 Database not created
- 94 Database bad: was being modified without output deferred, may not be accessed in mode n.
- 95 Database bad: Creation was in process (create again)
- 96 Database bad: Erase was in process (erase again)
- 253 Database enabled for indexing but third party indexing is not configured.

Exceptional Conditions

- 67 Third party indexes corrupted
- 68 Third party product indexes do not match product in System XL

Values Reserved for Third Party Indexing Functions [**Extensions**]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

DBCLOSE

DBCLOSE (base, dset, mode, status)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

1-3 Standard TurboIMAGE/XL modes

Modes Reserved for Third Party Indexing Functions [Extensions]

1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

-11 Bad base parameter
-21 Bad dset parameter
-31 Bad mode

Exceptional Conditions

63 DBG disabled; potential damage; only DBCLOSE allowed.

Values Reserved for Third Party Indexing Functions [Extensions]

3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

DBCNTROL

DBCNTROL (base, qualifier, mode, status)

base parameter: Same as current TurboIMAGE/XL

qualifier parameter: Same as current TurboIMAGE/XL

mode parameter:

- 1-6 Standard TurboIMAGE/XL modes
- 800 DBFIND mode 12 returns the prior list if the refinement results in 0 entries qualified.
- 801 DBFIND mode 12 no entries if the refinement results in 0 entries qualified. **[This is the default for SUPERDEX. If no records qualify, an error is returned, not the previous DBFIND qualifications.]**
- 802 Exposes the third party index data sets within the database. **[Only affects DBINFO]**
- 803 Hides the third party index data sets within the database. **[Only affects DBINFO. This is the default for SUPERDEX.]**

Modes Reserved for Third Party Indexing Functions [Extensions]

- 1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

Same as current TurboIMAGE/XL

Exceptional Conditions

Same as current TurboIMAGE/XL

Values Reserved for Third Party Indexing Functions [Extensions]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

DBPUT

DBPUT (base, dset, mode, status, list, buffer)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

- 1 Standard TurboIMAGE/XL DBPUT, with appropriate updates to any third party indexes

Modes Reserved for Third Party Indexing Functions [Extensions]

- 1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

Same as current TurboIMAGE/XL

Exceptional Conditions

Same as current TurboIMAGE/XL

Values Reserved for Third Party Indexing Functions [Extensions]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

list parameter: Same as current TurboIMAGE/XL

buffer parameter: Same as current TurboIMAGE/XL

DBDELETE

DBDELETE (base, dset, mode, status)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

- 1 Standard TurboIMAGE/XL DBPUT, with appropriate updates to any third party indexes

Modes Reserved for Third Party Indexing Functions [Extensions]

- 1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

Same as current TurboIMAGE/XL

Exceptional Conditions

Same as current TurboIMAGE/XL

Values Reserved for Third Party Indexing Functions [Extensions]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

DBUPDATE

DBUPDATE (base, dset, mode, status, list, buffer)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

- 1 Standard TurboIMAGE/XL DBPUT, with appropriate updates to any third party indexes

Modes Reserved for Third Party Indexing Functions [Extensions]

- 1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

Same as current TurboIMAGE/XL

Exceptional Conditions

Same as current TurboIMAGE/XL

Values Reserved for Third Party Indexing Functions [Extensions]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

list parameter: Same as current TurboIMAGE/XL

buffer parameter: Same as current TurboIMAGE/XL

DBINFO

DBINFO (base, qualifier, mode, status, buffer)

base parameter: Same as current TurboIMAGE/XL

dset parameter: Same as current TurboIMAGE/XL

mode parameter:

- 101 Defines type of access available for a specific item. No change from current TurboIMAGE/XL.
- 102 Describes specific data item. No change from current TurboIMAGE/XL.
- 103 Identifies all data items available in database and type of access allowed. No change from current TurboIMAGE/XL.
- 104 Identifies all data items available in specific data set and type of access allowed. No change from current TurboIMAGE/XL.
- 201 Defines type of access available for specific data set. No change from current TurboIMAGE/XL.
- 202 Describes specific data set. No change from current TurboIMAGE/XL.
- 203 Identifies all data sets available in database and type of access allowed. This mode does not show third party index files.
- 204 Identifies all data sets available which contain specified data items and type of access allowed. No change from current TurboIMAGE/XL.
- 301 Identifies paths defined for specified data set. No change from current TurboIMAGE/XL.
- 302 Identifies search item for specified data set. No change from current TurboIMAGE/XL.
- 401 Obtains information related to logging. No change from current TurboIMAGE/XL.
- 402 Returns information about ILR. No change from current TurboIMAGE/XL.
- 501 To check subsystem access to the database. No change from current TurboIMAGE/XL.
- 502 To retrieve critical item update settings for the database. No change from current TurboIMAGE/XL.

[Many terms, such as key or index, are misused, unclear, and confusing in the following 800 modes of DBINFO. SUPERDEX has assumed, in general, that terms such as key or item number refer to third party indexes and numbers, unless specified as TurboIMAGE/XL, in this appendix. There are no comments in this area about how SUPERDEX works. For a comprehensive description of how SUPERDEX processes the DBINFOS, refer to SUPERDEX User Manual, Section 5.]

- 801 Returns information on third party indexing package installed on this database (if any).
- qualifier: ignored
- buffer layout: 1-20 Product name (blank if none installed)
 21-25 Version number
 26-27 Date of current installation on a database (CALENDAR intrinsic format)
 28-29 Time of current installation on a database (CLOCK intrinsic format)
- 802 Returns the number of external as well as internal third party index files created for this database.
- qualifier: ignored
- buffer layout: 1 n = number of external files (may be 0).
 2 n = number of internal files (may be 0).
 3-10 Reserved for future use. Blanks for now.
- 803 Indicates if third party indexing is enabled for the database.
- qualifier: ignored
- buffer layout: 1 Binary zero if third party indexing is not enabled and binary one if third party indexing is enabled.
- 811 Describes the access available for a TurboIMAGE/XL data item or third party key.
- qualifier: 1-8 TurboIMAGE/XL data item name or number or third party key name or number.
 9-16 optional TurboIMAGE/XL data set name or number. When to be ignored, first character must be a blank or a ";".
- buffer layout: 1 TurboIMAGE/XL data item number or third party key number (10001 - 19999).

- 812 Describes a third party key. Also, used to determine third party key name and key type for third party keys that are not named the same as a TurboIMAGE/XL data item.
- | | | |
|----------------|-------|--|
| qualifier: | 1-8 | TurboIMAGE/XL data item name or number or third party key name or number. |
| | 9-16 | optional TurboIMAGE/XL data set name or number. When to be ignored, first character must be a blank or a ";". |
| buffer layout: | 1-8 | Third party key name |
| | 9 | Data type if simple third party key. Blank if a composite third party key. Valid data types: I, J, K, E, R, U, X, Z, P |
| | 10 | sub-item length. Must be zero for composite key. |
| | 11 | sub-item count. Must be zero for composite key. |
| | 12-13 | Reserved for future use. |
- 813 Identifies all data items available in database, including all third party keys, and type of access allowed. Externals are the same as DBINFO mode 103.
- 814 Identifies all data items available in a specific data set, including third party keys, and type of access allowed. External are the same as DBINFO mode 104.
- 821 Identifies all data sets available which contain the specified third party key and type of access allowed.
- | | | |
|----------------|---------|--|
| qualifier: | 1 | third party key name or number. |
| buffer layout: | 1 | Number of data sets which contain the third party key. |
| | 2-(n+1) | Data set number of the IMAGE data set which contains the key |
- 831 Identifies third party generic search keys for specified data set. Returns the item number of "G" or "B" type keys [**where "G" is generic only, "B" is both generic and multiple key**] as define in DBINFO mode 833.
- | | | |
|----------------|---------|--------------------------|
| qualifier: | 1-8 | data set name or number. |
| buffer layout: | 1 | n = number of keys |
| | 2-(n+1) | Item number of key |
- 832 Identifies third party single or multiple keyword search keys for specified data set. Returns the item number of "M" or "B" type keys [**where "M" is single or multiple keyword search key only, "B" is both generic and multiple key**] as define in DBINFO mode 833.

qualifier:	1-8	data set name or number.
buffer layout:	1 2-(n+1)	n = number of keys Item number of key
833 Describes a specific third party key		
qualifier:	1-8 9-16	set name or number of key item name or number of key
buffer layout:	1 2 3 4 5 6 7 8-27 28 6(n-1) +29 6(n-1) +30 6(n-1) +31 6(n-1) +32 6(n-1) +33 6(n-1) +34	<p>third party key number. If a negative value, data item can be updated. Third party keys are assigned item numbers of 10,001 and above.</p> <p>type of key. Valid key type: G (generic sorted), M (multiple key), and B (both generic sorted and multiple key) followed by 1 space.</p> <p>external key length (in bytes) [if SICOGNOS JCW = 1, length is net of extension; otherwise length includes the extension (extension = detail set: 4 bytes, master set: search item length in bytes).]</p> <p>IMAGE set number of first item in keyword group, or 0 if not member of a keyword group</p> <p>IMAGE item number of first item in keyword group, or 0 if not member of a keyword group</p> <p>IMAGE item number of third party key being qualified, or 0 if relative record being qualified instead of a third party key</p> <p>length of item in element 6 above (in bytes)</p> <p>an integer array describing this key's use of the standardized installation options, as shown on the table on next page</p> <p>n = number of key components</p> <p>IMAGE item number of component</p> <p>byte offset of item [if SICOGNOS JCW = 1, byte offset is ZERO relative; otherwise byte offset is ONE relative.]</p> <p>length of component in bytes</p> <p>data type of data in component: (I, J, K, E, R, U, X, Z, P) followed by a space</p> <p>IMAGE sub-item length of component</p> <p>IMAGE sub-item count of component</p>

Installation Options Table for DBINFO mode 833

<u>Array element</u>	<u>Value</u>
1 Case sensitivity	0 = Case insensitive 1 = Case sensitive
2 Parsing (treat entire argument as a key value)	0 = No parsing 1 = Parsing
3 Excluded words	0 = No words excluded 1 = Global words excluded 2 = Key-specific words excluded
4 Values: blanks, nulls, zeros	0 = Not indexed 1 = Indexed
5 Native Language Support	0 = Uses binary collating sequences 1 = Uses NLS collating sequences
6 Update of indexes	0 = DBPUT, DBDELETE, and DBUPDATE will update third party indexes. 1 = Indexes will not be updated until a third party maintenance utility is invoked to update the indexes.
7 Soundex	0 = No Soundex 1 = Soundex
8 Boolean searches within detail sets	0 = Common to record 1 = Common to detail chain
9-15 Reserved for future use	0 = Not used
16-20 Third party extensions	0 = No extension for key.

- 834 Describes all items grouped with the specified key.
- | | | |
|----------------|-----------|----------------------------------|
| qualifier: | 1-8 | set name or number of key |
| | 9-16 | Third party key name or number |
| buffer layout: | 1 | n = number of keys in this group |
| | 2(n-1) +2 | Set number of grouped key |
| | 2(n-1) +3 | IMAGE item number of grouped key |
- 901 To obtain the Native Language attribute of the database. No change from current TurboIMAGE/XL.

Modes Reserved for Third Party Indexing Functions [Extensions]

- 1nnn The ranges of modes 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

status parameter:

The status array may reference third party index keys and sets, as well as TurboIMAGE/XL items and sets.

Below are the minimum standards for exceptional conditions and error conditions. Note that where third party indexing products share the same value for the same purpose as TurboIMAGE/XL, this is shown as well. The values of 3nnn and -3nnn are reserved within TurboIMAGE/XL for status unique to the third party indexing products. The values may be assigned and controlled by the respective vendors.

Calling Errors

Same as current TurboIMAGE/XL

Exceptional Conditions

Same as current TurboIMAGE/XL

Values Reserved for Third Party Indexing Functions [Extensions]

- 3nnn The ranges of values 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standardized conditions.

list parameter: Same as current TurboIMAGE/XL

buffer parameter: Same as current TurboIMAGE/XL

DBERROR, DBEXPLAIN, DBLOCK, DBUNLOCK, DBBEGIN, DBMEMO, DBEND, DBXBEGIN, DBXEND, DBXUNDO, DBCALL

These intrinsics have no additional modes, argument tokens, or exceptional conditions within the standard. These intrinsics may have additional capabilities provided by the third party vendors that do not require modifications in the calling parameters. For example, DBERROR and DBEXPLAIN will appropriately handle exceptional conditions specifically related to third party indexing.

For each of these intrinsics, the range of mode 1000 to 1999 and -1999 to -1000 are reserved for third party indexing functions beyond the standardized functions.

For each of these intrinsics, the status conditions of 3000 to 3999 and -3999 to -3000 are reserved for reporting conditions beyond the standard TurboIMAGE/XL conditions.

APPENDIX B:**Internal structures**

SI-dataset structure

Between one and eight SI-datasets may be allocated for any database. Each dataset is a standalone detail set with the name **SI**, conditionally followed by the relative set number **1-7**.

The *root* SI-dataset contains the SUPERDEX definitions. It is normally named **SI**, but may alternately be named **SI0** if and only if a regular dataset named **SI** already exists.

If built by SIMAINT, the block size of each SI-dataset is equal to the database **BLOCKMAX** at the time the set is created.

The SI-datasets must appear consecutively in the dataset list in continuous numeric order starting with **SI** (i.e. **SI** or **SI0** immediately followed by **SI1**, **SI2**, etc.).

Additionally, write access must be granted to all user classes configured for write access to any SUPERDEX'ed dataset in the database.

SI-dataset capacity

The capacity of the SI-dataset is based on the space required to store the SI-indices, which depends on the actual value of the SI-keys. The calculation of the recommended SI-dataset capacity is performed by the SIMAINT utility under the assumption that the values of all SI-keys are different--the worst case condition. If there are many occurrences of the same SI-key values--especially for keyworded SI-paths--the actual space requirements may be considerably less than that calculated.

If you foresee this situation and there are B-trees for several SI-paths stored in a single SI-dataset, you may override the recommended capacity and specify one that is lower. It is safer, though, to use the recommended capacity and reduce it after the SI-indices have been generated. In fact, SIMAINT verifies that the SI-dataset capacities are sufficiently high based on the worst-case calculation.

❖ **It is also important to note that the SI-datasets capacity calculated by SIMAINT do not allow for future capacity changes in the datasets they index nor additional SI-paths that may be added at some later time. You may want to specify higher SI-dataset capacity to leave room for dataset capacity increases and additional SI-paths.**

SIMAINT initially generates B-trees that are optimized for space utilization. When performing heavy updates to the SI-indices, this optimization may be lost and the B-tree must expand. Therefore, it is recommended that 20 percent free space be left in each SI-dataset to accommodate this situation. The extra space utilized can always be regained by reorganizing all the SI-paths in the SI-dataset.

SI-item

The item named **SI** which is the only field in each SI-dataset.

The SI-item is built as a compound item by the SIMAINT.SUPERDEX.SYS utility program in the format *nX254*, where *n* is the subitem count as determined by the block size of the SI-dataset, with a maximum value of 16.

Alternately, as some software systems do not permit compound items, multiple individual items may be defined instead. These items should be named SI1 - SIn, with *n* the same as the subitem count.

Additionally, write access must be granted to all user classes configured for write access to any SUPERDEX'ed dataset in the database.

◆ **SI-index base is no longer supported with the SUPERDEX TPI version.**

SI-index

The SI-index is comprised of an SI-key followed by an SI-extension. The SI-index differs for SI-paths related to master and detail datasets.

For master sets, the SI-index consists of up to three SI-subkeys plus the IMAGE search field value, as shown:

----- SI-index -----			
----- SI-key -----			----- SI-extension -----
<i>SI-subkey 1</i>	<i>SI-subkey 2</i>	<i>SI-subkey 3</i>	<i>IMAGE search field value</i>

The search field may be specified as the last significant SI-subkey of a concatenated SI-key. In this case, the search field value is contained only once and is used for both selection and indexing.

For detail sets, up to four SI-subkeys are allowed, followed by a double-word relative record number:

----- SI-index -----				
----- SI-key -----				----- SI-extension -----
<i>SI-subkey 1</i>	<i>SI-subkey 2</i>	<i>SI-subkey 3</i>	<i>SI-subkey 4</i>	<i>Detail relative record number</i>

There are compression techniques that are used to save space when there are repeating SI-key values. For example, the SI-key value will not be stored twice if there are two records with the same SI-key and different master key values or relative record numbers.

Additionally, no duplicate SI-indices are stored. This means that if a detail record has a keyworded key, if a word is repeated in the field, there will only be one SI-index for that record.

SI-pointer

The SI-pointer consists of the last SI-index accessed plus one bit indicating whether the pointer is located in front of or after the current SI-index.

SI-subset

An extra data segment (XDS) that contains the results of a DBFIND performed in relational access mode. Ordinarily, the SI-subset contains only the SI-extensions that map the qualifying entries, as shown:

Master set:

<i>IMAGE search field value</i>

Detail set:

<i>relative record number</i>

The contents of the SI-subset are used for comparison with subsequent DBFIND calls.

If an SI-link is specified in addition to the SI-path in the DBFIND *item* parameter, the SI-subset also contains the value of the SI-link, as shown:

Master set:

<i>SI-link</i>		<i>IMAGE search field value</i>
----------------	--	---------------------------------

Detail set:

<i>SI-link</i>		<i>relative record number</i>
----------------	--	-------------------------------

In this case, both the SI-link and SI-extension are used for comparison. Also, the value of the SI-link is used to determine the sorting order when entries are returned.

APPENDIX C:**Maximum limits**

The following table identifies SUPERDEX's internal limits. Most limits are not checked, and results when exceeded are unpredictable.

SUPERDEX maximum limits

<i>Facility</i>	<i>Maximum Limit</i>
Number of Datasets per database with SI-paths	198
Number of SI-paths per database (with SI-index length under 30 words)	400
Number of SI-paths per database (with SI-index length of 99 words)	270
Number of SI-paths per dataset (with an SI-item length of 508 words)	22
Number of SI-paths per dataset (with an SI-item length of 1016 words)	53
SI-index length	127 words
SI-subkey length	63 words
Number of SI-indices per SI-dataset (with average SI-index length of 10 words)	102,400,000
Number of SI-indices per SI-dataset (with average SI-index length of 30 words)	68,266,000
Number of SI-indices per SI-dataset (with average SI-index length of 99 words)	20,686,000
Number of keywords per simple SI-key (or first SI-subkey when concatenated)	16
Number of items allowed in a grouped SI-path	32
Number of words in the keyword exclude path (with length of 4 words)	18,000
Maximum offset length (in bytes) for an SI-path	128
Maximum buffer size return by a DBGET using a <i>!list</i> parameter	2K

APPENDIX D:**Error and exceptional conditions**

SUPERDEX intrinsic error and exceptional conditions

SUPERDEX returns standard IMAGE condition codes and messages upon encountering an error or exceptional condition. These condition codes and messages describe SUPERDEX conditions that are equivalent to IMAGE conditions.

Also, because SUPERDEX uses standard IMAGE intrinsics to manage its B-tree structures, an error may indicate a problem in the SI-dataset rather than the dataset referenced by the *dset* parameter.

Some of the more common and noteworthy condition word values that may be returned by various intrinsics in accessing a SUPERDEX'ed base and what they mean are shown in the **SUPERDEX intrinsic error and exceptional conditions table** on the following pages.

SUPERDEX utility error and exceptional conditions

The **SUPERDEX utility error and exceptional conditions** table lists the various error messages that could be issued by the SIMAINT utility program, their meanings, and their corrective actions.

Program failures related to SUPERDEX

Programs that are run through the SUPERDEX SL or XL require certain capabilities and sufficient stack; otherwise, an error will occur. The **Program failures related to SUPERDEX** table lists these errors, their causes, and remedies.

SUPERDEX intrinsic error and exceptional conditions


<i>Type</i>	<i>Condition word/description</i>
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	-21 BAD PASSWORD DBOPEN Inconsistency in SI-definitions. Use SIMAINT,STRUCT against the database.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	-31 BAD (UNRECOGNIZED) DBFIND MODE: xxx DBFIND mode <i>1nn</i> or <i>2nn</i> The length imposed by the specified <i>mode</i> exceeds the length of the <i>argument</i> value. Specify a mode that does not exceed the <i>argument</i> length.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	-41 DBUPDATE WILL NOT ALTER A SEARCH OR SORT ITEM DBUPDATE Database was opened in <i>mode 2</i> and an update against one or more SI-key was attempted. Use a DBOPEN mode other than <i>mode 2</i> if updating SI-keys.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	-52 ITEM SPECIFIED IS NOT AN ACCESSIBLE SEARCH ITEM IN THE SPECIFIED SET DBFIND The specified <i>item</i> is neither an IMAGE key or a SUPERDEX SI-path. Use SIPATH to show the configured IMAGE keys and SUPERDEX SI-paths.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	-53 DBPUT LIST IS MISSING A SEARCH OR SORT ITEM DBPUT All IMAGE keys and SUPERDEX SI-keys are not included in the <i>list</i> parameter. Change the <i>list</i> to include all IMAGE keys and SUPERDEX SI-keys.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	-91 CANNOT ACCESS DATABASE DBOPEN The structure of the database does not match the SDX library. Be sure the correct library is being used. If so, restamp the database with SIMAINT.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	10 BEGINNING OF FILE DBGET <i>mode 16</i> After calling DBFIND <i>mode 100</i> or <i>200</i> and DBGET <i>mode 16</i> , the entry with the highest alphabetic SI-key in the dataset has been returned. Depends on the program design.
<i>Message</i> <i>Intrinsic</i> <i>Meaning</i> <i>Action</i>	11 END OF FILE DBGET <i>mode 15</i> After calling DBFIND <i>mode 100</i> or <i>200</i> and DBGET <i>mode 15</i> , the entry with the highest alphabetic SI-key in the dataset has been returned. Depends on the program design.

<i>Message</i>	14 BEGINNING OF CHAIN
<i>Intrinsic</i>	DBGET <i>mode</i> 5 or 15
<i>Meaning</i>	In <i>mode</i> 5, the entry with the lowest alphabetic SI-key that matches the specified DBFIND <i>argument</i> has been returned. In <i>mode</i> 15, the entry with the lowest alphabetic SI-key in the dataset has been returned.
<i>Action</i>	Depends on the program design.
<i>Message</i>	15 END OF CHAIN
<i>Intrinsic</i>	DBGET <i>mode</i> 6 or 16
<i>Meaning</i>	In <i>mode</i> 6, the entry with the highest alphabetic SI-key that matches the specified DBFIND <i>argument</i> has been returned. In <i>mode</i> 16, the entry with the highest alphabetic SI-key in the dataset has been returned.
<i>Action</i>	Depends on the program design.
<i>Message</i>	16 THE DATA SET IS FULL
<i>Intrinsic</i>	DBPUT
<i>Meaning</i>	Either the dataset referenced in the <i>dset</i> parameter or the SI-dataset that corresponds with the <i>dset</i> is full (the <i>dset</i> will be displayed in either case).
<i>Action</i>	Increase the capacity of the <i>dset</i> , or the SI-dataset.


SUPERDEX utility error and exceptional conditions

<i>Type</i>	<i>Description</i>
<i>Message</i>	17 THERE IS NO CHAIN FOR THE SPECIFIED SEARCH ITEM VALUE
<i>Intrinsic</i>	DBFIND
<i>Meaning</i>	<ol style="list-style-type: none"> 1. No entry exists that matches the specified <i>argument</i>. 2. An SI-index that has no corresponding data record was detected.
<i>Action</i>	<ol style="list-style-type: none"> 1. Even though an error is returned, if called in <i>mode</i> 1nn or 2nn, the internal SI-pointer is set and DBGET <i>mode</i> 15 and 16 may be used to retrieve the entries with SI-keys greater than and less than the specified <i>argument</i>. 2. Reorganize the suspected corrupt SI-path using SIMAINT.
<i>Message</i>	18 BROKEN CHAIN - FORWARD AND BACKWARD POINTERS NOT CONSISTENT
<i>Intrinsic</i>	DBGET
<i>Meaning</i>	Possible inconsistency in B-tree, due to program abort or system failure.
<i>Action</i>	Reorganize the SI-paths related to the suspected dataset, or all the SI-paths in the database.

<i>Message</i>	60 DATABASE ACCESS DISABLED
<i>Intrinsic</i>	DBOPEN
<i>Meaning</i>	The copy of SUPERDEX has expired, or the database has not been stamped by a new version of SIMAINT.
<i>Action</i>	If a current version of SUPERDEX is available, either demo or permanent, run SIMAINT against the database, otherwise delete the SI-item and SI-dataset(s) from the database.
<i>Message</i>	CAPACITY EXCEEDS MPE LIMIT - SI DATASET CANNOT BE CREATED
<i>Meaning</i>	The capacity calculated for a newly defined SI-dataset requires a worst-case dataset file that exceeds the MPE limits.
<i>Action</i>	Specify an additional SI-dataset(s) as necessary, or specify smaller capacity size.
<i>Message</i>	CAPACITY OF SI_n DATASET NOT SUFFICIENT - NO LOADING RECOMMENDED CAPACITY FOR SI_n: xxxxxxxx
<i>Meaning</i>	The capacity of the current SI-dataset is not high enough to accommodate the new SI-indices, so the SI-path configuration is saved but no SI-indices are generated.
<i>Action</i>	Change the capacity of the SI-dataset indicated to at least the recommended capacity, and run SIMAINT to populate the related SI-paths.
<i>Message</i>	COMPOUND ITEM NOT ALLOWED HERE
<i>Meaning</i>	A compound item was specified for an SI-subkey as other than the first SI-subkey
<i>Action</i>	Compound items may only be used as the first SI-subkey in an SI-key
<i>Message</i>	CREATOR ACCESS REQUIRED
<i>Meaning</i>	You are not logged on as the creator of the database you are attempting to access.
<i>Action</i>	Log on as the creator of the database and rerun the process.
<i>Message</i>	DEFINITIONS CANCELED FOR CURRENT DATASET
<i>Meaning</i>	\ was entered after defining an SI-path for a dataset, so any SI-paths defined for this dataset in the current run of SIMAINT are flushed
<i>Action</i>	None (status message only)
<i>Message</i>	EXTENSION FAILED - DATABASE IS OK
<i>Meaning</i>	An error was detected during processing, but the database was not damaged.
<i>Action</i>	Correct the condition that caused the error and rerun SIMAINT.
<i>Message</i>	EXTENSION FAILED - PLEASE GO TO BACKUP OF DATABASE
<i>Meaning</i>	An error was detected during the extension phase of SIMAINT and the database is damaged.
<i>Action</i>	Correct the error condition displayed, restore the database from backup, and rerun SIMAINT.
<i>Message</i>	FILE ERROR ACCESSING SI-DATASET
<i>Meaning</i>	An MPE file system error was detected while accessing the newly-created SI-dataset.
<i>Action</i>	Correct the error condition displayed, and rerun SIMAINT.

<i>Message</i>	FILE ERROR ACCESSING NEW ROOT FILE
<i>Meaning</i>	An MPE file system error was detected while accessing the newly-rebuilt database root file.
<i>Action</i>	Correct the condition that caused the error, and rerun SIMAINT.
<i>Message</i>	FILE ERROR ACCESSING OLD ROOT FILE
<i>Meaning</i>	An MPE file system error was detected while accessing the current database root file.
<i>Action</i>	Correct the condition that caused the error, and rerun SIMAINT.
<i>Message</i>	GROUPING NOT ALLOWED FOR THIS SI-PATH
<i>Meaning</i>	An independent SI-path was configured as grouped.
<i>Action</i>	This configuration is illegal.
<i>Message</i>	NONEXISTENT DATASET
<i>Meaning</i>	A dataset name was specified that does not exist in the database.
<i>Action</i>	Enter ? for a list of datasets.
<i>Message</i>	ILLEGAL OPTION
<i>Meaning</i>	An unrecognized suffix was specified on either the database, dataset or the SI-path name.
<i>Action</i>	Valid dataset suffixes are /1 - /7, /D, and /R. Valid SI-path suffixes are /A, /P, /B, /D, /G, /K, and /R.
<i>Message</i>	ILLEGAL OPTION - NO CORRESPONDING SI-DATASET EXISTS
<i>Meaning</i>	Either the referenced SI-dataset (specified via <i>dataset/n</i>) does not exist, or the SI-datasets do not appear in numerical consecutive order in the database.
<i>Action</i>	The assigned SI-dataset does not exist and SIMAINT.SUPERDEX.SYS was not executed, so the SI-dataset must be built by hand. In the latter case, the SI-datasets must appear in sequential order.
<i>Message</i>	INPUT ERROR READING DATASET
<i>Meaning</i>	A file system error was detected when reading the dataset.
<i>Action</i>	Print a copy of the error tombstone and call Bradmark Technical Support  .
<i>Message</i>	INPUT SORTLIB: TOO MANY RECORDS
<i>Meaning</i>	The <i>average number of keywords</i> specified in configuring a keyworded SI-path or <i>average number of indices</i> for a custom SI-path is not high enough.
<i>Action</i>	Use SIMAINT to reorganize the SI-path, and specify a higher average number of keywords or indices.

<i>Type</i>	<i>Description</i>
<i>Message</i> <i>Meaning</i> <i>Action</i>	INVALID SI-DATASET 1. For a database with one or more SI-paths, the root SI-dataset has been corrupted. 2. For a database with no SI-paths, a regular dataset named SI already exists. 1. Erase the root SI-dataset using a utility (e.g. via DBGGENERAL) or delete all its entries, and redefine all SI-paths for the database. 2. Configure SUPERDEX with a root SI-dataset of SI0 (refer to <i>Section 3 Configuration/Establishing SI-indices</i> for details).
<i>Message</i> <i>Meaning</i> <i>Action</i>	ITEM NOT TYPE U OR X - SI-PATH CANNOT BE KEYWORDED The specified item may not be configured as the first SI-subkey in a keyworded SI-path because it is not alphanumeric (data type U or X). Numeric data types are not supported for keywording.
<i>Message</i> <i>Meaning</i> <i>Action</i>	ITEM NOT IN DATASET The specified item does not exist in the current dataset. Enter ? for a list of items in the current dataset.
<i>Message</i> <i>Meaning</i> <i>Action</i>	LENGTH CONFLICT IN GROUP The item specified for grouping is of a longer length than other items in the group. Configure the longest SI-key in the group first.
<i>Message</i> <i>Meaning</i> <i>Action</i>	MAXIMUM OF 199 SETS EXCEEDED Creating the configured SI-dataset(s) would cause the database to have more than 199 dataset (99 for non-Turbo IMAGE databases). Specify fewer SI-datasets.
<i>Message</i> <i>Meaning</i> <i>Action</i>	NO SI-PATHS DEFINED FOR SPECIFIED DATASET A dataset was specified with a suffix of /D or /R , but no SI-paths are related to the dataset. /D and /R are only allowed on datasets with existing SI-paths.
<i>Message</i> <i>Meaning</i> <i>Action</i>	ODD NIBBLES NOT ALLOWED The specified item is of data type P and its subitem length is odd. The subitem length for type P items must be even.
<i>Message</i> <i>Meaning</i> <i>Action</i>	SI-PATH ALREADY EXISTS An SI-path with the specified name already exists for the dataset. Specify a unique SI-path name or append /G to group the SI-path.
<i>Message</i> <i>Meaning</i> <i>Action</i>	SI-PATH DOES NOT EXISTS An SI-path that does not exist in this database was specified with either /D , /G , or /R . Specify the name of an existing SI-path.
<i>Message</i> <i>Meaning</i> <i>Action</i>	SORTLIB ERROR: TOO MANY INPUT RECORDS The <i>average number of keywords</i> specified in configuring a keyworded SI-path or <i>average number of indices</i> for a custom SI-path is not high enough. Reorganize the SI-path, and specify a higher average number of keywords or indices.

<i>Message</i>	TOO MANY SI-PATHS DEFINED FOR <i>dataset</i>
<i>Meaning</i>	More SI-paths have been specified for a related dataset than the maximum limit.
<i>Action</i>	Refer to the table in the <i>Appendix C Maximum Limits</i> , or call Bradmark Technical Support for information on how to define more  .
<i>Message</i>	TYPE CONFLICT IN GROUP
<i>Meaning</i>	The item specified for grouping is of a different data type than the already existing SI-path.
<i>Action</i>	All items in a group must be of the same data type.
<i>Message</i>	WARNING: ILR ENABLED
<i>Meaning</i>	Non-critical message indicating the ILR is enabled for the database when SIMAINT is executed, and therefore processing may be slower.
<i>Action</i>	In the future, disable ILR prior to executing SIMAINT.
<i>Message</i>	WARNING: LOGGING ENABLED
<i>Meaning</i>	Non-critical message indicating that logging is enabled for the database when SIMAINT is executed, and therefore processing may be slower.
<i>Action</i>	In the future, disable logging prior to executing SIMAINT.
<i>Message</i>	WARNING! INCONSISTENCY DETECTED. RERUN WITH ,STRUCT
<i>Meaning</i>	An inconsistency between the database structure and the SUPERDEX configuration was detected.
<i>Action</i>	Run SIMAINT,STRUCT against the database.
<i>Message</i>	WARNING! OPTION IGNORED, PREVIOUS DEFINITION RETAINED
<i>Meaning</i>	The specified SI-dataset option does not match the existing SI-dataset option for the dataset.
<i>Action</i>	Once an SI-path has been defined for a dataset, the SI-dataset option can not be modified. To change the SI-dataset, the existing SI-path(s) must be deleted and reconfigured with the new SI-dataset option specified.

Program failures related to SUPERDEX

<i>Type</i>	<i>Error/Description</i>
<p><i>Message</i></p> <p><i>Cause</i></p> <p><i>Remedy</i></p>	<p>PROCESS QUIT; PARM = 62 (PROGRAM ERROR #18) Exceeds available SUPERDEX table. Call Bradmark Technical Support. ☎</p>
<p><i>Message</i></p> <p><i>Cause</i></p> <p><i>Remedy</i></p>	<p>PROCESS QUIT; PARM = 63 (PROGRAM ERROR #18) Exceeds available SUPERDEX table. Call Bradmark Technical Support. ☎</p>
<p><i>Message</i></p> <p><i>Cause</i></p> <p><i>Remedy</i></p>	<p>PROCESS QUIT; PARM = 64 (PROGRAM ERROR #18) Exceeds available SUPERDEX table. Call Bradmark Technical Support. ☎</p>

SUPPLEMENT:

TRANSACT interface

Description

There is no special interface necessary for TRANSACT and SUPERDEX users. The TRANSACT Laboratory has published the following article describing how to utilize TRANSACT with SUPERDEX. This article is reprinted with permission from Hewlett-Packard.

Third Party Indexing Made Simple With HP Transact

Generic key search to TurboIMAGE data has been the single most requested TurboIMAGE enhancement. As of MPE/iX release 4.0, a TurboIMAGE interface to Third Party Indexing (TPI) has been added to provide end-users and software vendors simplified access to the indexing products from Bradmark Technologies and Dynamic Information Systems Corporation. Over the past year, HP has also worked closely with these vendors to improve TPI access from one of HP's high level programming languages, HP Transact. Using HP Transact, programmers can easily access the key functionality offered by the TPI products using existing HP Transact I/O verbs. This article summarizes how an HP Transact programmer can configure and user TPI retrieval methods.

CONFIGURING TPI

Configuring TPI retrievals is an important step. TPI index configurations control the behavior of your HP Transact verbs. You can set up an index that performs one or more types of Third Party Indexing. This section gives you some basic information for choosing and configuring a TPI index for your HP Transact application environment.

TPI indexing methods are defined in this article as either generic key searches or keyword searches. For both methods of indexing, one or more data items can be associated to form a key.

The first method of indexing, generic key searches, concatenates all the data item values named in the key to form a single key value for each data entry. Generic key searches allow you to qualify data entries based on all or part of the key value using wildcards and relational operators.

The second indexing method, keyword searches, takes one or more data items of text (values are groups of characters separated by spaces or punctuation) and creates a key for each unique value. Keyword searches allow you to retrieve all records that contain a given value in any of the items defined by the index.

Generic Key Searches

Generic key search indexes are useful if your HP Transact programs are serially reading all of the records in a TurboIMAGE data set to perform the following retrievals:

- Selection of data by a partial key. For example, you can use a partial key search to find all the clients whose surname starts with the characters "SM".
- Selection of data by a range of key values. For example, you can use range searches to find all orders with a shipping weight between 1 and 5 pounds.
- Selection of large amounts of data in sorted order. For example, you can retrieve all orders between 1 and 5 pounds in ascending order by weight.

Generic key searches allow the use of the MPE/iX wildcards "@", "?" and "#" to qualify an entire key value starting from first character of the left. For example, the key value "SM@" finds the names "SMITH" and "SMYTHE". Additionally, the key value "PN##A" finds a part number "PN12A" and "PN13A". Wildcards can be combined with the "<", "<=", ">", ">=" relational operators to refine searches. The key value ">SM@" finds all records with a value of "SN" through the highest key values. The data entries from a generic key search are returned in ascending order, so you do not need to sort by key value. More detailed information on generic key searches can be found in the documentation for each TPI product.

Keyword Searches

Keyword searches are useful if you are currently reading all of the records in a TurboIMAGE data set to perform this retrieval:

- Selecting data entries based on a combination of values (or parts of values) within one or more items of text data. For example, a value of "market@" finds all data entries that contain either the value "market" or "marketing" anywhere within one or more data items of text.

When you create a keyword search type of TPI index, every value in a text data item is a key value. All of the wildcards and relational operators defined in a generic key search can be used to qualify each value within the text. Additionally, four Boolean operators ("AND", "NOT", "TO", and "OR") can be used to refine the selection criteria.

Given the following text for the "Text-Item":

Data Entry #	Text-Item
1	Part 1234 arrived 1 day late to ABC Corporation.
2	Part 5555 arrived 3 days early to ABC Corporation.
3	Part 1234 arrived 1 week early to XYZ Corporation.

you may selectively retrieve data entries based on values within the item "Text-Item". For the three data entries listed above, you can retrieve all records which contain the value "early" by specifying the key value "early". You can also retrieve all records which contain either "day" or "days" by specifying a key value of "day@". Finally, you can find the data entries which contain both the value "1234" and "early" specifying the key value "1234 and early". Please note that keyword searches can be configured to ignore case. More information on keyword searches can be found in the documentation for each TPI product.

Creating the Index

After choosing a type of indexing, you should assemble the following information to help you create your indexes:

- The types of indexing needed.
- The data items that construct key values for each index you wish to define within each database.
- Values of text that you do not wish to include for keyword searches (like "a" or "the").
- The vendor specific commands and nomenclature needed to configure the indexes.

Indexes are then created by running OMNIUTIL for OMNIDEX, or by running SIMAINT for SUPERDEX. Each utility will ask you to identify the database name, data set name, key item name, a list of data items which construct the key, key type, and many other options. Specific configuration examples may be found in the documentation for both TPI products. For the purpose of the HP Transact Examples (used later) we have configured an index on item1 of set1 within the dbtpi database using each TPI configuration utility.

USING TPI

TPI indexes are used in two modes. First, TPI indexes can enhance existing HP Transact verbs by activating TPI wildcards and operators. Second, TPI indexes can enhance retrievals when calling TurboIMAGE intrinsics from HP Transact.

Using TPI Through HP Transact Verbs

To use TPI with HP Transact, use TPI retrievals from HP Transact verbs whenever possible. HP Transact verbs perform most of the chores involved in application programming. Figure 1 (below) shows the OUTPUT verb with TPI. No program modifications were necessary for TPI, but new key values are allowed as noted in the program comments. Other HP Transact verbs may be used in a similar manner and require no program modifications.

```

<<#####>>
<<  Figure 1: OUTPUT VERB WITH TPI  >>
<<#####>>
<< Perform chained read function.  Use "A@" for  >>
<< the key value or ">A<D" for the key value and receive  >>
<< a Third Party Indexing assist for partial keys and key  >>
<< ranges.  >>
<<  >>
<< This example uses OUTPUT, but the FIND, GET, REPLACE,  >>
<< and DELETE verbs work in the same manner.  >>
<<  >>
<< Third parties also have extensions that allow:  >>
<<  * Pop up pick lists for selection.  >>
<<  * Pre-selection for batch programs.  >>
<<  >>
<< *****  >>

system tpil,
    base = dbtpi(";",1);

<< ----- >>
<< These data items could be defined in a dictionary.  >>
<< ----- >>
    define(item) item1      x(20):
                   item2      x(20):
                   item3      x(20);

<< ----- >>
<< List items for retrievals.  >>
<< ----- >>
    list item1:item2:item3;

get-key:

    data item1 ("Enter Partial Key: ");
    if (item1) = "/" then exit;
    set(key) list(item1);

    output(chain) set1,list=(item1,item2,item3);

end tpil;

```

Programming Tips for TPI through Transact Verbs

Figure 1 illustrates that TPI works with no changes to many existing HP Transact applications. If, however, one of the conditions below applies, you may wish to make some modifications.

- If you are using database verbs within a PERFORM= option, HP Transact can cause the TPI product to re-qualify records. If verbs like FIND, DELETE, or REPLACE result in poor performance when used with the PERFORM= option, adding a SORT= option can improve performance.
- If the search criteria value is longer than the argument, you can allow larger search criteria by making one of the following changes:
- Create a TPI index with a unique name (which is not named in the database schema) and DEFINE a corresponding HP Transact item with an appropriate length. Then you may reference the unique index name in the SET(KEY) verb preceding your search.
- Use DEFINE(ITEM) to increase the storage length for your key item. If you retrieve data into the key item using a database verb, your key item must be the last item in the "LIST=" item list option. Placing the key item anywhere else in the "LIST=" option will cause your data to be misaligned and placed into the wrong data items.
- If the application performance depends on HP Transact to optimize a serial read operation such as FIND(SERIAL), you should modify the application to perform a chained read operation like FIND(CHAIN).
- If the application source is not available, you can use pop up pick lists or batch mode pre-processing record selection options which are available from the TPI vendors.

Using TPI Through TurboIMAGE Intrinsic Calls

TPI contains some features that are not accessible through HP Transact verbs. If you wish to manage the qualified list of entries using DBFIND mode 12 and mode 13, you need to call TurboIMAGE directly. Figure 2 (below) is provided to show you the steps necessary to successfully perform a keyword retrieval (mode 12) so you can make a direct comparison with the complexity of Figure 1.

```

<<#####>>
<<  Figure 2: STEPS TO PERFORMANCE KEYWORD RETRIEVAL  >>
<<#####>>
<< Perform a keyword retrieval using TurboIMAGE verbs. This >>
<< is more difficult, but allows access to all Third >>
<< Party Indexing modes for DBFIND and DBGET. >>
<< >>
<< Though keyword searches are available from mode 1 DBFIND,>>
<< this example shows an explicit request for a keyword >>
<< search. >>
<< ***** >>

system tpi2,
      base = dbtpi(";",1);

```

```

<< ----- >>
<< These data items could be defined in a dictionary. >>
<< ----- >>

define(item) dbbuffer      x(60):
        item1      x(20) = dbbuffer(1):
        item2      x(20) = dbbuffer(20):
        item3      x(20) = dbbuffer(40);

<< ----- >>
<< Local variables used for calling TurboIMAGE directly. >>
<< ----- >>

define(item) mode          i(4):
        argument      x(50):
        stat          10 i(4):
        itemlist      x(18):
        errorlength   i(4);

<< ----- >>
<< List items for keyword retrievals. >>
<< ----- >>
list dbbuffer:mode:stat:itemlist:argument:errorlength;

<<*****>>
<< Get Key value and set up Keyword Retrieval. >>
<<*****>>
get-key:

data argument ("Enter Pattern Match");
if (argument) = "/" then exit;

let (mode) = 12;
move (itemlist) = "ITEM1;";
let (stat) = 0;

<<*****>>
<< Locate records by keyword using DBFIND mode 12. >>
<<*****>>
proc DBFIND(base,set(set1),(mode),(stat),(itemlist),(argument));
if (stat) <> 0 then
do
perform database-error;
go to get-key;
doend;

<<*****>>
<< Retrieve all qualified records using DBGET mode 25. >>
<< * * * * * >>
<< Note: item1, item2, and item3 are contiguous within >>
<< the item dbbuffer. >>
<<*****>>
move (itemlist) = "ITEM1,ITEM2,ITEM3;";
let (mode) = 25;

```



```

while (stat) = 0
do
  proc DBGET(base, set(set1),(mode),(stat),
             (itemlist),(dbbuffer),(argument));
  if (stat) = 0 then
    display(table) item1:item2:item3
  else if (stat) <> 15 then
    perform database-error;

  doend; << while stat = 0 >>e
go to get-key;

<< ***** >>
<< Process a database error. >>
<< ***** >>
database-error:
  move (argument) = " ";
  proc DBERROR((stat),(argument),(errorlength));
  display "Database Error selecting records: ";
  display argument, col=3, stat;
  return;

end tpi2;

```

SUMMARY

Through the efforts of Bradmark Technologies and Dynamic Information Systems Corporation, Third Party Indexing is available to many existing HP Transact systems. You can now perform generic key searches, range retrievals, and keyword searches through HP Transact with no programming changes.

AVAILABILITY

The fully integrated TurboIMAGE and TPI package is available on MPE/iX release 4.0 or later. TPI vendors also have versions of OMNIDEX and SUPERDEX which emulate the TPI externals for systems running a version of MPE/iX older than 4.0 and all versions of MPE V. Please contact Dynamic Information Systems Corporation and Bradmark Technologies for software which is compatible with your version of MPE/iX or MPE/V.

POSTSCRIPT

This article which has been provided by the HP Transact team is an introduction of TPI to the HP Transact community. For a full list of options and distinguishing features, please refer to the sales information and documentation for each product.

SUPERDEX INDEX

<p style="text-align: center;">i</p> <p>- 3-10 4-6</p> <p style="text-align: center;">!</p> <p>! List 2-27 4-31 5-18</p> <p style="text-align: center;">#</p> <p># 2-13 3-10 4-10</p> <p style="text-align: center;">/</p> <p>/A 3-18,22</p> <p>/B 3-7,18,23,25</p> <p>/D 3-16,18 6-4,7</p> <p>/G 2-18,21 3-6,18,23,25,26 6-7</p> <p>/K 2-9 3-5,18,21,23,25,26</p> <p>/n 3-16</p> <p>/N 3-28</p> <p>/P 3-18,22</p>	<p style="text-align: center;">/R</p> <p>..... 3-16,18 6-4,7</p> <p style="text-align: center;">;</p> <p>; 4-5</p> <p style="text-align: center;">?</p> <p>? 2-13 3-10 4-10</p> <p style="text-align: center;">@</p> <p>@ 2-12 3-10 4-10,15</p> <p>@ list SIUSER procedure 5-31</p> <p style="text-align: center;">[</p> <p>[..... 4-5,15,18</p> <p style="text-align: center;">~</p> <p>~ 4-5</p> <p style="text-align: center;"><</p> <p><<>> 2-12 4-10</p> <p><= 1-10 2-16 4-6,12</p> <p><> 1-10</p>
--	--

.....	2-16	Batch	
.....	4-6,12	SIMAINT	
=		3-33
==		Binary Large Objects	
.....	4-6	Independent SI-key	
>		2-26
>=		Blank SI-key	
.....	1-10	3-7
.....	2-16	BLOBS	
.....	4-6,11	Independent SI-key	
4		2-26
4GL,	SIQTP5	Boolean operations	
4GL,	SIQUIZ5	2-22
A		4-5
Account Structure		Boolean operators	
.....	1-6	4-17
Active SI-subset		,	
.....	4-18	4-17
Adding entries		+	
.....	4-2	4-16
AND		1-11
.....	1-11	4-18,22,24,26
.....	2-22	AND	
.....	4-5	4-16
DBFIND		Combined	
.....	4-16	4-18
Multiple value retrieval		DBFIND	
.....	4-16	4-21
AND NOT		NOT	
.....	1-11	4-17
.....	4-5	OR	
DBFIND		4-17
.....	4-12,17	BRW	
Approximate match retrieval		1-18
.....	1-10	B-tree	
.....	2-14	Standalone	
ASK Date		2-26
.....	1-7	Vs automatic master	
ASK Date Format		1-12
.....	1-17	Business Report Writer	
ASK Date SI-key		1-18
Defining		Byte Lengths	
.....	3-22	1-9
Autodefer		Byte Offset	
.....	5-6	SI-path	
Automatic master		3-20
.....	3-4	Byte Offsets	
Replacing with SI-path		1-9
.....	1-17	Byte Offsets and Lengths	
Vs B-tree		3-4,14,19
.....	1-12	Index Substrings	
B		2-3,5,7,12,14
Backup SI-subset		Restrictions	
.....	4-18	3-7

Substring field	3-22
Definition	3-7
Restrictions	3-7
Vs sorted chains	1-12
Capacity	
SI-dataset	3-28
Chain count	5-10
Chains	
Vs SI-indices	1-12
Combined Boolean operators	4-18
Compatibility	1-5
DBFIND Mode 10	1-5
DBFIND Mode 2nn	1-6
DBINFO Modes	1-6
Explicit Locking	1-5
Implicit Locking	1-5
SICOMPAT JCW	1-6
Utility Programs	
Location	1-6
Complex DBFIND	4-19
Compound item	1-13
.....	3-4
.....	4-4
Restrictions	3-7
Compound SI-key	
Keywording	2-10
Concatenated SI-key	1-9
.....	2-5,25
.....	3-4
.....	4-4
DBFIND	4-13
Byte Lengths	4-13
Defining	3-19
Keyworded	
.....	3-22
Restrictions	3-7
Vs sorted chains	1-12
Condition word	
DBGET	4-30
Condition word 14	5-17
Condition word 15	5-17
Condition word 17	4-28
.....	5-7,9
Condition word -21	4-28
Condition word -52	4-28,32
Corresponding entries	
DBFIND	4-23
Creating	
SI-dataset	3-13
SI-item	3-13
Current path	
DBFIND	4-28
Effect	5-15
DBGET	4-32
Effect	5-19
Effects of SI-intrinsics	4-33
Custom Separators	
Keywords	1-7
Custom SI-key	1-11
.....	2-25
.....	3-7
Defining	3-26
SIUSER procedure	2-25
.....	4-2
.....	5-30
Customization string	3-10

D	
Data type conversion	2-25
Data type K	4-6
Data types	1-17
	4-6
	5-13
Database	
Defining	3-16
Disabling	3-2
Enabling	3-2
Database maintenance	6-3
Dataset	
Defining	3-16
Defining associated SI-datasets	3-17
Date reformatting	2-25
	4-2
DBCONTROL	
SI-intrinsic	5-3
DBDELETE	5-5
DBDELIX	2-27
	4-3
SIDRIVER	6-52
SI-intrinsic	5-4
DBERASE	
Recovery	5-6
SIDRIVER	6-52
SI-intrinsic	5-5
DBFIND	
AND	4-16
AND NOT	4-12,17
Argument	
*	4-25
@@	
	4-20
	4-6,15
Boolean operators	4-21
Complex	4-19
Compound item	4-14
Concatenated SI-key	4-13
Byte Lengths	4-13
Corresponding entries	4-23
Current path	4-28
Effect	5-15
Examples	4-7
Generic retrieval	4-10
Greater-than	4-11
Group SI-key	4-13
Independent SI-key	2-27
Indexed access	4-6
Keyword SI-key	4-14
Less-than	4-12
Mode 10	
Pre-TPI Compatibility	1-5
Mode 2nn	
Pre-TPI Compatibility	1-6
Modes	2-13
	4-5
Multiple calls	2-24
Multiple databases	4-23
Multiple datasets	4-21
Multiple sets/bases	
Projection	4-24
Multiple SI-paths	4-21

Not-equal	4-12	Mode 811	5-21
OR	4-17	Mode 812	5-22
Partial retrieval	4-10	Mode 813	5-22
Pattern matching	4-12	Mode 814	5-22
Projection		Mode 821	5-22
Multiple sets/bases	4-24	Mode 831	5-23
Range	4-12	Mode 832	5-23
Relational access	4-15	Mode 833	5-24
Restrictions	5-14	Mode 834	5-25
SI-intrinsic	5-7	Modes	
SI-pointer	4-28	Pre-TPI Compatibility	1-6
Effect	5-15	SI-intrinsic	5-20
Summary	4-4	DBOPEN	
Super-grouped SI-key	4-13	SI-intrinsic	5-26
DBGENERAL interface	6-4	DBPUTIX	2-27
DBGET		SI-intrinsic	4-3
Condition word	4-30	SIDRIVER	6-52
Current path	4-32	SI-intrinsic	5-27
Effect	5-19	Default characters	3-10
Independent SI-key	2-27	Deferring indexing	
Retrieving entries	4-30	SIMAINT	3-28
SI-intrinsic	5-16	Deleting	
SI-pointer	4-32	SI-path	3-18
Effect	5-19	Deleting entries	4-2
DBINFO		Disabling a Database	3-2
Mode 801	4-33	Dynamically-joined indices	2-24
Mode 802	5-21		
Mode 803	5-21	E	
		Effects of SI-intrinsics	
		Current path	4-33
		SI-pointer	4-33
		Efficiency	

.....	2-4,6
Generic retrieval 2-13
Keywording 2-10
.....	3-5
Partial retrieval 2-13
Relational access 2-23
Enabling a Database 3-2
.....	3-2
Enhancements,SIMAINT4	
Enhancements,SUPERDEX4	
Entry count 4-20
Entry points	
SIMAINT 6-5
Error handling	
Compatibility 1-17
Errors 6-2
Exceptional conditions 6-2
Exclusion words	
Defining 3-28
Keywording 3-5,8
Extension phase	
SIMAINT 3-28
F	
FASTRAN 4-34
First-on-chain 5-10
Function Keys	
SUPERDEX(Program) 6-15
G	
General Description 1-2
Generic retrieval 1-9
.....	2-12
.....	3-10
.....	4-5
DBFIND 4-10
Efficiency 2-13

Greater-than	
DBFIND 4-11
Retrieval 1-10
.....	2-16
Grouped retrieval 1-10
.....	2-18
Grouped SI-key 3-6
.....	4-4
Compound item 3-4
DBFIND 4-13
Defining 3-23
Keyworded 3-26
Restrictions 3-7
H	
High Speed OR'ing 4-27
I	
ILR 5-6
IMAGE	
Access 3-3
Intrinsics enhancements 5-2
Path 3-4
Independent SI-key 1-11
.....	2-26
.....	3-7
Accessing 4-3
Binary Large Objects 2-26
BLOBS 2-26
Defining 3-17,27
Index Substrings 3-4
Index value	
SIUSER procedure 5-31
Indexed access 4-31

Vs relational access	1-16	KWEXCLUDE SI-path	2-9
.....		
Vs Relational access	4-5	Length	2-9
.....		
Indexing phase		3-5,8
SIMAINT	3-29	Maximum limits	2-10
.....		
Infix Notation	1-13	Minimum number of characters	2-9
.....		
.....	4-15	Keyword retrieval	1-9
.....	5-28	2-9
INFO		Compound item	2-10
SIMAINT	3-10,11	Keyword SI-key	3-5,11
.....		4-4
Installation	3-2	5-13
.....		DBFIND	4-14
Invoking SIMAINT	6-6	Defining	3-21
.....		
Item		Grouped	3-26
Parameter	3-3	
.....		Restrictions	3-7
SI-path,SI-link	4-22	
.....		Keywords	
J		Custom Separators	1-7
JCW		KWEXCLUD	
SIEXTLEN	5-17	Default File	3-9
.....		
.....	6-6,13	File	3-8
SISSETLINK	5-10,16,17	
.....		MPE file	6-2,40
JCW SICOMPAT	1-6	
.....		KWEXCLUDE SI-path	2-9
Job stream		3-5,8
SIMAINT	6-11	6-40
.....		Defining	3-17,28
K		L	
K datatype	4-6	Last-on-chain	5-10
.....		
Keyword		Less-than	
Average number of indices	2-9	DBFIND	4-12
.....		
Average number of keywords	3-5	Retrieval	1-10
.....		2-16
Custom separators	3-10	List	
.....		!	4-31
Duplicate words in SI-key	2-10	
.....			
Efficiency	2-10		
.....			
.....	3-5		
Exclusion words	3-5,8		
.....			
Hyphenated words	2-10		
.....			

Locking	
Explicit	
Pre-TPI Compatibility	1-5
Implicit	
Pre-TPI Compatibility	1-5
SIREPAIR	6-40
SITEST	6-40
Logging	3-4
	5-6
	M
Master dataset	
Access Vs detail dataset	1-12
Multiple keys	2-3
Mode 801	
DBINFO	4-33
	5-20
Mode 802	
DBINFO	5-21
Mode 803	
DBINFO	5-21
Mode 811	
DBINFO	5-21
Mode 812	
DBINFO	5-22
Mode 813	
DBINFO	5-22
Mode 814	
DBINFO	5-22
Mode 821	
DBINFO	5-22
Mode 831	
DBINFO	5-23
Mode 832	
DBINFO	5-23
Mode 833	
DBINFO	5-24
Mode 834	
DBINFO	5-25
Modes	
SITEST	6-41
MPE file	
KWEXCLUD	6-2,40
MPE flat file	2-26
Multiple @ signs	4-10
Multiple criteria	2-24
Multiple databases	
DBFIND	4-23
Multiple datasets	
DBFIND	4-21
Multiple field retrieval	
Application	2-24
Multiple keys	2-3
Multiple sets/bases	
DBFIND	
Projection	4-24
Multiple SI-keys	1-9
	N
Native Language Support	1-18
	2-8
	3-27
	4-35
Negative values	
Sorting	2-8
NLS	1-18
	2-8
	3-27
	4-35
NOT	2-22
Not-equal	
DBFIND	4-12
Retrieval	2-16

	O			
OR				
.....		1-11		
.....		2-22		
.....		4-5		
DBFIND				
.....		4-17		
	P			
Parameter				
Item				
.....		3-3		
Partial retrieval				
.....		1-9		
.....		2-12		
.....		3-10		
DBFIND				
.....		4-10		
Efficiency				
.....		2-13		
Pattern matching				
DBFIND				
.....		4-12		
PowerHouse				
.....		1-18		
.....		4-34		
PowerHouse Date				
.....		1-7		
PowerHouse Date Format				
.....		1-17		
PowerHouse Date SI-key				
Defining				
.....		3-22		
Pre-TPI Compatibility				
.....		1-5		
DBFIND Mode 10				
.....		1-5		
DBFIND Mode 2nn				
.....		1-6		
DBINFO Modes				
.....		1-6		
Explicit Locking				
.....		1-5		
Implicit Locking				
.....		1-5		
JCW SICOMPAT				
.....		1-6		
Utility Programs				
Location				
.....		1-6		
Procedure				
SIUSER				
.....		5-30		
Projection				
.....		4-24		
				5-11
DBFIND				
Multiple sets/bases				
.....		4-24		
SI-link				
.....		4-21		
PROTOS				
.....		4-34		
	Q			
QTP				
.....		1-18		
.....		4-34		
Qualified entries				
.....		5-8		
Qualifying entries				
.....		4-4		
Overview				
.....		1-16		
QUICK				
.....		1-18		
.....		4-34		
QUIZ				
.....		1-18		
.....		4-34		
	R			
Range				
DBFIND				
.....		4-12		
Retrieval				
.....		1-10		
.....		2-16		
Real numbers				
.....		4-7,11,12		
Recovery				
DBERASE				
.....		5-6		
Redefining SI-path				
.....		6-4		
Related detail datasets				
.....		2-20		
Relational access				
.....		3-4		
.....		4-5,7,26,31		
DBFIND				
.....		4-15		
Efficiency				
.....		2-23		
Multiple criteria retrieval				
.....		1-11		
.....		2-22		
Multiple databases				
.....		1-11		
.....		2-24		
Multiple datasets				

.....	1-11	SI-counter	4-31
.....	2-24	5-18
Multiple fields		Definition	1-15
.....	1-11	SI-dataset	
.....	2-24	Capacity	3-28
Vs Indexed access		Creating	3-13
.....	1-16	Defining multiple	3-17
.....	4-5	Definition	1-15
Relational Table Overflow (see High Speed		SI-definitions	
OR'ing)		Definition	1-15
.....	4-27	SIDRIVER	6-52
Reorganizing		SI-extension	
SI-path		Definition	1-14
.....	3-18	Independent SI-key	2-26
.....	6-4	SIEXTLEN JCW	
Restrictions		2-6
DBFIND		3-15,20
.....	5-14	5-17
SI-key		6-6,13
.....	3-7	SI-index	
Reverse Polish Notation		Definition	1-14
.....	1-13	Managing explicitly	4-3
.....	4-15	SI-index base	
.....	5-28	Definition	1-15
RPN		SI-indices	
.....	4-15	6-2
S		Reading	4-31
Sample applications		4-31
.....	1-8	Reading multiple indices	4-31
Semi-colon		Vs chains	1-12
.....	4-5	SI-intrinsic	5-1
Sequential Access		DBCONTROL	5-3
.....	4-32	DBDELIX	5-4
Serial Access		DBERASE	5-5
.....	4-32	DBFIND	
Shared Access			
SIMAINI			
.....	1-6		
Shared SIMAINI			
.....	3-15		
SI			
Definition	1-14		
SI-chain			
Definition	1-14		
Repositioning	4-30		
SICOMPAT JCW	1-6		
SICOUNT	6-46		
JCW	3-30		

.....	5-7	5-6
DBGET 5-16	6-9
DBINFO 5-20	Default Progress Interval 3-30
DBOPEN 5-26	Deferring indexing 3-28
DBPUTIX 5-27	Dialog phase 3-14
Definition 1-15	Entry points 6-5
SITRANSLATE 5-28	Example 3-31
SI-item		Full Screen 6-14
Creating 3-13	In batch 3-33
Definition 1-15	INFO 3-10,11
SI-key 2-5	Input rules 3-14
Defining 3-3	Blank value 6-5
Blank value 3-19	Job stream 6-11
Definition 1-14	LIST entry point 3-12
Extracting 2-25	6-10
Length 2-4	Options 6-5
Restrictions 3-7	Performance 3-14
SI-key Substrings		6-5
Restrictions 3-7	Running in batch 6-13
SI-key value		SCHEMA entry point 6-11
Determining 4-2	SHARED 3-15
SI-link 2-7	Shared Access 1-6
.....	3-4	SI-indices	
.....	4-21,23,24,25	Establishing 3-13
.....	5-11	STRUCT entry point 6-13
Definition 1-15	SUPERDEX.SYS 3-13
Projection 4-21	Restrictions 3-13
SIMAINT 3-3	3-13
.....	6-5	XL= 5-31
Access requirements 3-14	SIMAINT, Enhancements 1-4
.....	6-5	SIMAINT, single pass of dataset 1-4
DBLOAD entry point		Simple SI-key 2-3

.....	3-4	Access requirements	6-40
Defining	3-18	Invoking	6-44
SI-path	3-4	Locking	6-40
.....	5-7	Specifying Input	6-44
Byte Offset	3-20	Specifying request before update	6-45
Defining	3-3,18	SISSETLINK JCW	5-10,16,17
Definition	1-14	SI-subkey	2-5
Deleting	3-18	Definition	1-14
.....	6-8	SI-subset	Active
KWEXCLUDE	6-40	Backup	4-18
Name	3-3	Definition	1-14
Numbers	5-10	SITEST	6-40
Redefining	6-4	Access requirements	6-40
Reorganizing	3-18	Locking	6-40
.....	6-4,7	Modes	6-41
Starting Position	3-20	Running in batch	6-44
SIPATH	6-35	TREETEST	6-41
Windows	1-7	SITRACE	6-50
SI-path,SI-link		SITRANSULATE	Example
Item	4-22	SI-intrinsic	5-29
SI-pointer	2-14	SIUSER procedure	5-28
.....	5-7	@ list	5-31
DBFIND	4-28	3-7
Effect	5-15	4-2
DBGET	4-32	5-30
Effect	5-19	Custom SI-key	2-25
Definition	1-15	Index value	5-31
Effects of SI-intrinsics	4-33	Sorted chains	2-5,7
SIQTP, Uses QTP UDC5			
SIQUIZ, Uses QUIZ UDC5			
SIREPAIR	6-40		

SUPERDEX

Fourth-Generation Language Interface

User Manual

Version 4.2

All updates to or derivatives of the SUPERDEX™ computer software provided herein are copyrighted and may not be copied except for archival purposes, to replace a defective copy, or for program error verification by Licensee. Copyrighted material may not be copied onto any media (e.g. magnetic tape, paper tape, disc memory cartridges, read-only memory, etc.) for any other purposes. The authorization to duplicate copyrighted materials hereunder shall not be construed to grant the Licensee or Licensee's customer the right to use copyrighted SUPERDEX material in any manner other than which is provided in this agreement or otherwise approved in writing by Dr. Wolfgang Matt or Bradmark Technologies, Inc.

© 1988, 1993, 2000 Bradmark Technologies, Inc.

All rights reserved

Printed in the U.S.A. (July 1993)

(Updated January 1994)

(PDF October 2000)

IMAGE, TurboIMAGE, and TurboIMAGE/iX are trademarks of Hewlett-Packard Company

PowerHouse, PDL, QTP, QUICK, and QUIZ are registered trademarks of Cognos Incorporated

Speedware is a registered trademark of Speedware Corporation

SUPERDEX is a trademarked product name of Bradmark Technologies, Inc. for the SI-IMAGE package developed and implemented by Dr. Wolfgang Matt

About this manual

In writing this manual, we have assumed that you have working knowledge, although not internal knowledge, of IMAGE and the HP3000, as well as the PowerHouse module or modules you need to interface to SUPERDEX.

All references to IMAGE in this manual and throughout the SUPERDEX package also apply to TurboIMAGE and TurboIMAGE/iX unless otherwise noted.

This manual is arranged in the following format:

Section 1 provides an *Overview* of the SUPERDEX Fourth-Generation Interface, its capabilities, and benefits.

Section 2 describes how SUPERDEX interfaces to Cognos' QUICK, QUIZ, and QTP, and the various access methods that are available for retrieving entries using SUPERDEX.

Section 3 describes how SUPERDEX interfaces to Speedware's Speedware, and the various access methods that are available for retrieving entries using SUPERDEX.

Table of contents

Section 1: Overview	1-1
Section 2: PowerHouse	2-1
PowerHouse Overview	2-2
How the QUICK Interface works	2-3
How the QUIZ interface works.....	2-4
How the QTP interface works	2-5
Access methods	2-6
Configuration	2-11
Installation.....	2-12
Data dictionary changes for QUICK	2-13
QKGO file changes for QUICK interface	2-15
QUICK interface.....	2-17
Modifying screens for SUPERDEX access	2-18
Qualifying entries for retrieval	2-20
SIPARMS item.....	2-21
SICOUNT item.....	2-23
Sample QUICK screen	2-24
QUIZ interface	2-27
Invoking SIQUIZ.....	2-28
Modifying source files for SUPERDEX access.....	2-29
Qualifying entries for retrieval	2-31
SIQUIZ Commands	2-32
SIDEF command	2-33
SIPATH command	2-35
Other SIQUIZ commands.....	2-38
Notes on QUIZ commands	2-40
Sample source files.....	2-41
QTP interface	2-43
Invoking SIQTP	2-44
Error and exceptional conditions	2-45
Section 3: Speedware I 1	3-1
Speedware Overview	3-2
Access methods.....	3-3
Installation.....	3-7
Data Dictionary changes for Designer	3-8
Designer Interface.....	3-11
Qualifying entries for retrieval.....	3-21
SIPARMS item.....	3-22
SICOUNT item.....	3-24

SECTION 1:

Overview

Overview

The SUPERDEX Fourth-Generation Language interface was designed and implemented so that users of Fourth-Generation Languages can easily utilize SUPERDEX's lookup and retrieval capabilities.

Currently, this interface is documented for Cognos' PowerHouse package, and Speedware Corp.'s Speedware package.

The interface allows for complete access to the SUPERDEX product. It can be used to increase the performance and flexibility of 4GL online applications, along with often time-consuming reporting. The features and capabilities described in the SUPERDEX User Manual can be used in conjunction with the SUPERDEX 4GL Interface to enhance existing applications along with new applications.

This release of the SUPERDEX 4GL Interface (Version 4.2) utilizes TurboIMAGE/iX's Third-Party Indexing Interface (TPI). This interface in TurboIMAGE/iX allows TurboIMAGE/iX to load and call SUPERDEX intrinsics directly, removing the need for system administrators to set up any special command files or User-Defined Commands to establish the SUPERDEX intrinsic library.

SECTION 2:

PowerHouse

This section discusses the use of the SUPERDEX 4GL Interface to Cognos' QUIZ, QUICK, and QTP.

Chapter 1 PowerHouse Overview

Function Describes the basic function of the each interface and the access methods.

Chapter 2 Configuration

Function Describes the configuration of the PowerHouse dictionary.

Chapter 3 QUICK Interface

Function Describes the SUPERDEX access methods in the QUICK interface.

Chapter 4 QUIZ Interface

Function Describes the SUPERDEX access methods in the QUIZ interface.

Chapter 5 QTP Interface

Function Describes the SUPERDEX access methods in the QTP interface.

Chapter 6 PowerHouse Error and exception conditions

Function Lists the various Error and Warning messages that could be issued by QUICK, SIQUIZ, or SIQTP, their meanings, and corrective actions.

CHAPTER 1:

PowerHouse Overview

This chapter overviews the interface to QUICK, QUIZ, and QTP.

The first three parts describe *How the QUICK interface works*, *How the Quiz interface works*, and *How the QTP interface works*.

The last part overviews the various *Access Methods* available through SUPERDEX's Fourth-Generation Language Interface and PowerHouse.

How the QUICK interface works

Installation

The QUICK interface requires that a dummy manual master dataset called **SIPROC** and its elements be added to the data dictionary for each database that will be accessed via SUPERDEX (for multiple bases, each **SIPROC** file is suffixed with a number). Each PowerHouse dictionary that wishes to utilize the SUPERDEX interface must have the **SIPROC** file described.

These modifications are minimal and allow QDESIGN to compile cleanly.

Qualifying and retrieving entries

The QUICK interface works by reinterpreting the **SEQUENTIAL** option of the **GET** verb to perform sorted indexed-sequential SUPERDEX access rather than an IMAGE serial read. To accomplish this, only minor additions to QUICK screens are required, and no changes to default statements and procedures generated by QDESIGN are required.

In each QUICK screen, the statement **FILE SIPROC DESIGNER** to declare the dummy **SIPROC** file as a **DESIGNER** file and **ACCESS VIA SIPARMS** to define the access method, are added.

Also, a **POSTPATH** procedure is written which accesses the **PATH** designated for sequential access by QDESIGN and sets up the parameters for SUPERDEX access using standard **LET** statements. The dataset to access is defined in the **SIDSET** parameter; the SI-path is defined in the **SIPATH** parameter; and the **SIMODE** parameter specifies how entries should be qualified and whether they should be returned in ascending or descending order. The search value, partial or generic key, range of values, or multiple search values are specified with the desired operators in the **SIARG** parameter.

Because most of SUPERDEX's capabilities are accomplished by embedding multiple values and conditional, relational, and Boolean operators in the search argument, it is normally sufficient to simply vary the value of the **SIARG** and use an **SIMODE** of 1.

Entries are by default returned in ascending sorted sequential order, and may alternately be returned in descending order by altering the **SIMODE**. Other **SIMODE**s may be used to additionally retrieve entries that are alphabetically higher or lower than the **SIARG** value and to perform other specialized functions, such as returning the alphabetically highest or lowest entry in the dataset.

After defining the **SIPARMS**, the **POSTPATH** procedure calls **GET SIPROC USING SIPARMS OPTIONAL NOGENERIC**.

The default QDESIGN-generated **FIND** procedure remains unchanged: SUPERDEX automatically redefines the access performed by the **SEQUENTIAL** option to be an indexed-sequential read via SUPERDEX rather than an IMAGE serial read. After entries are retrieved, the qualifying number of entries is returned in the **SICOUNT** item when **SIMODE** 1 is used.

How the QUIZ interface works

Installation

User-Defined Commands (including PHUDC.CURRENT.COGNOS), menus, job streams, etc. should be changed to run SIQUIZ with an **INFO** parameter, to define the location of QUIZ, which then automatically invokes QUIZ.

A front-end program called **SIQUIZ** is provided in SUPERDEX.SYS. It is run instead of QUIZ, and defines the QUIZ source file to **USE** and interprets two new SUPERDEX QUIZ interface commands which specify the SUPERDEX access to perform.



The QUIZ interface requires no modifications to the data dictionary.

Qualifying and retrieving entries

In SUPERDEX's QUIZ interface, entries are qualified and retrieved using the new **SIPATH** command, which specifies the dataset, SI-path, and search argument. It is used in place of or in addition to the **SELECT** command. The **CHOOSE** command is not used to access SI-paths, but continues to work as always on IMAGE search fields.

As in the QUICK interface, the search argument specified (in the **SIPATH** command) may contain partial and generic key values, ranges, multiple values, and conditional, relational, and Boolean operators.

For cases in which the user is prompted for the search criteria, the prompt is included in the **SIPATH** command, and the user's response may include multiple values and operators. The **SIDEF** command may be used like QUIZ's **DEFINE** command to define and initialize variables for use in **SIPATH** commands.

For retrievals involving multiple SI-paths, multiple **SIPATH** commands are used. Retrievals involving multiple datasets also require the use of multiple **SIPATH** commands, with each specifying the dataset to access. Entries are unconditionally returned in ascending sorted sequential order.


How the QTP interface works

Installation

Installation is the same as for the QUIZ interface.

User-Define Commands (including PHUDC.CURRENT.COGNOS), menus, job streams, etc. should be changed to run SIQTP with an INFO parameter, to define the location of QTP, which then automatically invokes QTP.

A front-end program called **SIQTP** is provided in SUPERDEX.SYS. It is run instead of QTP, and defines the QTP source file to **USE** and interprets two new SUPERDEX QTP interface commands, which specify the SUPERDEX access to perform.

 **The QTP interface requires no modifications to the data dictionary.**

Qualifying and retrieving entries

The QTP interface uses the same commands and operates identically to the QUIZ interface.

Access methods

Multiple keys in master and detail datasets

QUICK: A **POSTPATH** procedure is written to set up the parameters for SUPERDEX access, including the SI-path to access, and **GET SIPROC USING SIPARMS OPTIONAL** is called. When the retrieval is done via **GET...SEQUENTIAL** in the QDESIGN-generated **FIND** procedure, an indexed-sequential read is performed by SUPERDEX rather than a serial read.

QUIZ and QTP: The **SELECT** command is replaced by or works with an **SIPATH** command, which defines the SI-path to access and the value to search for. This causes QUIZ or QTP to perform an indexed-sequential read via SUPERDEX rather than a serial read. The **CHOOSE** command is still used for access by IMAGE search fields but is not used for SUPERDEX access.

Concatenated keys containing multiple fields

QUICK: The *SIARG* item in the data dictionary is redefined as necessary to represent the SI-subkeys in the concatenated SI-key, their lengths, and their data types. The values for each SI-subkey are defined as separate fields in the **POSTPATH** procedure.

QUIZ and QTP: The QUIZ and QTP interfaces support only concatenated SI-keys that contain SI-subkeys of data types X or U (or concatenated SI-keys whose first SI-subkey is of type X or U, allowing access via the specification of the first SI-subkey only). The value of each SI-subkey in the concatenated SI-key is assigned to a separate variable, which may be done by hard-coding or prompting using the **SIDDEF** command. Then, these values are concatenated together in the **SIPATH** command.

Sorted sequential retrieval


Entries are unconditionally returned in sorted sequential order for entries qualified in indexed access mode. In relational access mode, an SI-link may be specified in the *SIPATH* parameter to enforce sorting order.

QUICK: By default, entries that match the specified *SIARG* are returned in ascending order. By adding 2000 to the *SIMODE*, entries are instead returned in descending order. All the entries in a dataset may be read in ascending or descending sorted order by using *SIMODE* 1100 or 3100, respectively.

QUIZ and QTP: Entries are unconditionally returned in ascending sorted sequential order. To read all the entries in a dataset in ascending order, specify an *SIARG* of **@**. If a different sort is required, simply access the file serially and add the **SORT** command. This is typically much faster.

Keyword retrieval

Keyworded SI-paths may be accessed through QUICK, QUIZ, and QTP without restriction.

 **Keyword retrieval is available only in the SUPERDEX II package.**

Generic and partial key retrieval

QUICK: Partial key access can be performed in SUPERDEX'ed QUICK using two different methods, which are similar to the generic access methods used for KSAM files in QUICK.

The first is to use an *SIMODE* of 1 and specify the partial key value appended with an @ as the *SIARG* (e.g. **HEWL@**). This will locate all entries that match on the significant characters followed by anything.

The second method is to specify the value in the *SIARG* without an @ but vary the *SIMODE* based on the length of the value in the *SIARG*. For example, an *SIARG* containing the partial key **ROLA** would dictate *SIMODE* 102 or -104 (100 plus the number of words or bytes, respectively, in the value).

Generic key retrieval is accomplished by embedding the ? matchcode in the *SIARG*, which holds the place of any alphanumeric character. For example, the *SIARG* **L?TTER** would locate "LETTER" and "LITTER"; by appending an @ (**L?TTER@**), "LETTERMAN", "LITTERBUG", and "LOTTERY" would also be located.

QUIZ and QTP: Partial key retrieval is performed by specifying the partial key value appended with an @ in the *SIPATH* command. For generic key retrieval, one or more ? matchcodes are embedded in the search value, each one acting as a place holder for any alphanumeric character.

Approximate match retrieval

QUICK: Approximate match retrieval is performed by using an *SIMODE* that specifies how many characters in the *SIARG* SUPERDEX should match on, which is typically the length of the value specified. If no matching entry exists, the nearest matching entry is returned.

The *SIMODE* also dictates whether the internal SI-pointer in the B-tree should be set before or after the matching or nearest matching entry, and whether entries should be returned in ascending or descending order.

For example, an *SIARG* containing the value **UNITED** would dictate *SIMODE* 1103 or -1106, both of which would cause SUPERDEX to match on the entire value. The *SIMODE* is calculated as 100 plus the number of words or bytes (negated if bytes), plus or minus 1000 to dictate ascending order retrieval. Using these *SIMODEs*, the SI-pointer would be set before the matching or nearest matching entry. Any entry beginning with "UNITED" would be included in the retrieval.

If *SIMODE* 3103 were instead specified (adding 3000 instead of 1000 for descending order), any "UNITED" entry would be excluded, since the SI-pointer is set before the matching entry and entries are being returned in descending order. With an *SIMODE* of 3503 or -3506 (500 plus the number of words or bytes, plus or minus 3000 for descending order retrieval), the SI-pointer would be set after the matching or nearest matching entry, and the "UNITED" entries would now be included.

Note that for compatibility with QUICK, if SUPERDEX does not find a matching entry and returns the nearest matching entry, **ACCESS <> OK** is returned, but the internal SI-pointer is still set. In this case, the program should be written to ignore this error.

QUIZ and QTP: Approximate match retrieval is not supported from either QUIZ or QTP.

Greater-than or equal-to, less-than or equal-to, and range retrieval

QUICK: These retrievals are performed by embedding special operators in the *SIARG* for *SIMODE* 1 or 10.

Greater-than-or-equal-to retrieval is accomplished by prefixing the search value with the **>=** operator (e.g. **>=1000**), less-than-or-equal-to retrieval uses the **<=** operator as a prefix, and not-equal-to retrieval uses the **<>** operator. The **<>** operator can also appear after another value in the same *SIARG* to perform an Boolean AND NOT operation (e.g. **SUPER@<>SUPERDEX**).

Range retrievals are performed by using the **>=** and **<=** operators in combination. For example, a range search to find all the entries with amounts between 500 and 1000, inclusive, is specified with the *SIARG* **>=500<=1000**.

By default, entries are returned in ascending sorted order, but may be returned in descending order by adding 1000 to the *SIMODE*.

QUIZ and QTP: These retrievals are performed by embedding the same operators as described for the QUICK interface in the search criteria. They may either be specified by the user or hard-coded in the program. Entries are unconditionally returned in ascending order. If descending order is desired, add a ***SORT item D*** statement in the QUIZ or QTP program.

Grouped retrieval



grouped retrieval is available only in the SUPERDEX II package.

QUICK: Whenever the group is referenced by its SI-path name in the *SIPATH* parameter of **SIPROC**, all SI-keys that form the group are unconditionally searched.

Grouped SI-paths are accessed in the same way as non-grouped SI-paths--the only difference is in the configuration of the SI-path.

There may be some ambiguity in searching by an SI-key in a grouped SI-path whose item length is shorter than the group length and which is therefore padded with spaces. For example, if CITY, an X16, and STATE, an X2, are grouped together with an SI-key length of 8 words (to accommodate CITY), an *SIARG* of **CA** would find not only all entries in the state of "California" but also those in the cities of "CALABASAS" and "CARLSBAD". To resolve this ambiguity, use *SIMODE* 1 or 21 and pad the *SIARG* with enough trailing spaces to cover the full SI-key length.

QUIZ and QTP: Whenever the group is referenced by its SI-path name in the **SIPATH** command, all SI-keys that form the group are unconditionally searched.


Super-grouped retrieval

 Super-grouped retrieval is available only in the SUPERDEX II package.

QUICK: To reference a super-grouped SI-path, use the IMAGE master set as the **PRIMARY** file. Quick will search all paths related to the super-grouped SI-path in the master and detail datasets, but will only display the master entries. To display the detail entries, reference the detail dataset as **DETAIL** or **SECONDARY** on the **FILE** statement.

QUIZ and QTP: To reference the master dataset, list it as the first file in the **ACCESS** statement. To reference the detail datasets, use the **LINK TO** option on the **ACCESS** statement. Whenever the super-group is referenced by its SI-path name in the **SIPATH** command, all SI-keys that form the group are unconditionally searched.

Relational access: multiple criteria retrieval

 Relational access retrieval capability using criteria is available only in the SUPERDEX II package.

QUICK: Boolean operations are accomplished by embedding the appropriate operators in the search value in the *SIARG*. The *SIMODE* can be either 1 or 12. In either case, a semicolon (;) must be suffixed to the *SIARG*. If *SIMODE* 1 is used the *SIARG* must be prefixed with a tilde (~).


For example, a search for all part descriptions that contain both the words PAPER and CLIP would be specified with an *SIARG* of **PAPER and CLIP;** for *SIMODE* 12 or **~PAPER and CLIP;** for *SIMODE* 1. To find all invoices that are unpaid or canceled, the *SIARG* would be **UNPD or CANC;** for *SIMODE* 12; and to find all entries in California and not Los Angeles, **~CA not "LOS ANGELES" ;** using *SIMODE* 1.

QUIZ and QTP: Boolean operations may be specified in either of two methods, based on whether search values are hard-coded in the program or prompted for.

If hard-coded, the same syntax as in the QUICK interface is used.

If the user is instead prompted for the search criteria, the values may be specified using simpler syntax and operators. For example, to perform the search for all part descriptions that contain both the words PAPER and CLIP, the user could specify **PAPER+CLIP**. To find all entries in California and not Los Angeles, the user would specify **CA-"LOS ANGELES"**, and to find all invoices that are unpaid or canceled, the user would enter **UNPD,CANC**. If necessary, parentheses can be used to enforce a special order of evaluation.


Relational access: multiple fields, sets, bases

 **Relational access retrieval capability using multiple fields, sets, and bases is available only in the SUPERDEX II package.**

QUICK: Multiple **GET SIPROC USING SIPARMS OPTIONAL** statements are performed in succession, with one **GET** per SI-path. If performing retrievals against multiple fields or sets within a database, the *SIDSET* and *SIPATH* parameters are varied to specify the desired access. If performing retrievals against multiple bases, **SIPROC** is suffixed with a number that specifies the base referenced in the data dictionary.

In the preceding example, four **GETs** would be performed in succession: one against the PAID-FLAG SI-path in the ORDER-HEADER set in the ORDERS base (referenced by **SIPROC1**), another against the ORDER-TOTAL-AMT SI-path, another against the AVG-DAYS-TO-PAY SI-path in the CUSTOMERS set, and the last against the dataset and SI-path of the same name but in the ORHIST database (referenced by **SIPROC2**).

QUIZ and QTP: Multiple **SIPATH** commands are used, with one for each SI-path, which specify both the SI-path and dataset to access.

 **Retrievals against multiple SIPATHs can only be issued against the database of the primary file.**

As in QUIZ and QTP, it is required that if entries are retrieved from more than one dataset that the sets be **LINKED TO** and related by IMAGE paths.

Custom indexing

QUICK: The custom SI-path is referenced in the *SIPATH* parameter and the custom SI-key value is specified in the *SIARG* parameter.

QUIZ and QTP: The custom SI-path and custom SI-key value are specified in the **SIPATH** command.

CHAPTER 2:

Configuration

This chapter describes the method for installing and configuring the SUPERDEX PowerHouse interface.

Refer to the **SUPERDEX User Manual** for information about installing SUPERDEX, establishing the SI-item and SI-dataset(s), defining SI-paths, and establishing B-trees and SI-indices.

This chapter assumes that you have already loaded SUPERDEX on your system, as described in the separate SUPERDEX loading instructions.

The first part discusses the *Installation* procedure for the PowerHouse interface.

Next, the *Data dictionary changes for QUICK* are described. If the QUICK interface will not be used, these changes are not required.

The last part describes the *OKGO file changes for QUICK*. These changes are also not required if the QUICK interface will not be used.

Installation

Once the SUPERDEX installation tape has been loaded, as described in the separate SUPERDEX loading instructions, only a couple of steps are necessary for completing the SUPERDEX installation.

Some of these steps are one-time operations, while others are required for future creation of users, accounts, programs, etc. The installation procedures may include:

- changes to the COGNOS User-Defined Commands
- data dictionary changes for the QUICK interface
- QKGO file changes for the QUICK interface

Changes to the COGNOS User-Defined Commands

The UDCs (such as PHUDC.CURRENT.COGNOS) that run QUIZ, and QTP need to be changed to run SIQUIZ and SIQTP. As an example, if your QUIZ and QTP programs are located in the CURRENT group of the COGNOS account, change the :RUN commands as follows:

```
:RUN SIQUIZ.SUPERDEX.SYS;INFO="PROCLOC=.CURRENT.COGNOS"  
:RUN SIQTP.SUPERDEX.SYS;INFO="PROCLOC=.CURRENT.COGNOS"
```

Be sure to use these same commands in menus, job streams, command files, etc.

Data dictionary changes for QUICK

The following changes can be applied to the PDL source file. The changes are displayed in the order of a PDL file since this is the new dictionary.

The QUICK interface requires a dummy manual master dataset to be added to the data dictionary for each database that will be accessed via SUPERDEX. To do so:

1. Obtain a current PDL source file.
2. Add **ELEMENT** statements for the elements that will be used in subsequent **RECORD** definitions:

```

ELEMENT SIPARMS          CHARACTER      SIZE 200
ELEMENT SIDSET           CHARACTER      SIZE 16
ELEMENT SIPATH           CHARACTER      SIZE 34
ELEMENT SIMODE           CHARACTER      SIZE 5      &
    DATATYPE INTEGER SIGNED SIZE 2
ELEMENT SIARG            CHARACTER      SIZE 148
ELEMENT SIARG1           NUMERIC        SIZE 10      &
    DATATYPE INTEGER SIGNED SIZE 4
ELEMENT SIARG2           CHARACTER      SIZE 144
ELEMENT SICOUNT          NUMERIC        SIZE 10      &
    DATATYPE INTEGER SIGNED SIZE 4

```

3. Add the **RECORD** definition for each **FILE** statement:

```


RECORD SIPROC                                &
    OPEN "SIPROC"                             &
ORGANIZATION MASTER                          NOCREATE
    ITEM SIPARMS
        BEGIN STRUCTURE
            ITEM SIDSET
            ITEM SIPATH
            ITEM SIMODE
            ITEM SIARG
        END STRUCTURE
    ITEM SICOUNT

INDEX SIPARMS
    SEGMENT SIPARMS
.
.

```

SIARG may be further redefined to support concatenated SI-keys (which contain more than one SI-subkey), for example:

```
ITEM SIARG
  BEGIN STRUCTURE
    ITEM SIARG1
    ITEM SIARG2
  END STRUCTURE
```

 For each **IMAGE FILE** (multiple databases), define a separate **SIPROC** Record, suffixing each **SIPROC** Record name with an integer starting with 1 (e.g. **SIPROC1**, **SIPROC2**, etc...). Additionally, each **SIPROC(n)** Record will access the same name of "**SIPROC**" on the **OPEN** parameter, NOT "**SIPROC(n)**". For example:

```
RECORD SIPROC1      &
OPEN "SIPROC"
```

-
4. Use PDL to compile the file into the new data dictionary.

QKGO file changes for QUICK interface

The QUICK interface requires that the expression size in the QKGO file be at least 200. With later releases of PowerHouse, the expression size default has been increased to above the 200 level, therefore there are no modification necessary to the QKGO file.

If you are using an older PowerHouse version, or would like to check your QKGO file:

1. Invoke the SETQKGO UDC:

SETQKGO

2. The HP3000 QKGO Construction and Maintenance Screen is displayed. Specify

F

in the ACTION field and press **RETURN**.

3. Then, specify the QKGO filename (usually QKGO) and press **RETURN**.

4. Enter

3

in the ACTION field and press **RETURN**.

5. The Execution-time parameter values screen is displayed. If the Expression size is less than **200**, specify

3

in the ACTION field and press **RETURN**.

6. Then, specify the value

200

and press **RETURN**.

7. Then, press **F8** to return to the first screen.

8. Back on the HP3000 QKGO Construction and Maintenance Screen specify

UR

in the ACTION field and press **RETURN**.

9. Then, press **F8** to exit.

CHAPTER 3:

QUICK interface

This chapter discusses SUPERDEX's interface to QUICK. It is assumed that you have:

- loaded the tape according to the separate SUPERDEX loading instructions,
- installed the PowerHouse interface,
- enhanced the data dictionary,
- and assured that the *expression size* in the QKGO file is at least 200.

The first part describes the process of Modifying screens for SUPERDEX access in order to access entries using SUPERDEX.

The next part explains the differences in Qualifying entries for retrieval between the QUICK interface and SUPERDEX's DBFIND intrinsic.

Part three describes the SIPARMS Item, which is used to define the parameters for SUPERDEX access.

Next, the SICOUNT item is discussed, which is used to return the number of qualifying entries.

The last part gives a before and after view of Sample QUICK screens modified for SUPERDEX access.

Modifying screens for SUPERDEX access

The QUICK interface reinterprets the **SEQUENTIAL** option of the **GET** verb to perform sorted indexed-sequential SUPERDEX access, along with SUPERDEX relational access, instead of an IMAGE serial read. Only minor additions to QUICK screens are required to accomplish this. No changes to the default statements and procedures generated by QDESIGN are required.

For each QUICK screen that requires SUPERDEX access, the following statements must be added:

```
FILE SIPROC DESIGNER
ACCESS VIA SIPARMS
```

Then, a small **POSTPATH** procedure must be written which accesses the *SIPATH* designated for access and sets up the parameters for SUPERDEX access (*SIPARMS*):

```
PROCEDURE POSTPATH
BEGIN
  IF PATH = 2
  THEN BEGIN
    LET SIDSET = "CUSTOMERS;"
    LET SIPATH = "CUSTOMER-NAME;"
    LET SIMODE = 1
    LET SIARG = "ACME MANUFACTURING "
    GET SIPROC USING SIPARMS OPTIONAL
    IF NOT ACCESSOK
    THEN
      ERROR "No records were found matching key values."
    END
  END
END
```

The *SIPARMS* are defined as follows:

SIDSET	name of the dataset containing entries
SIPATH	name of the SI-path to utilize
SIMODE	defines the method of SUPERDEX access
SIARG	the search argument

The values for *SIPARMS* are documented fully for reference in the *SIPARMS item* section later in this chapter.

In the example above, the customer "ACME MANUFACTURING " is being searched for in the CUSTOMERS dataset via the SI-PATH *CUSTOMER-NAME*. Note that a blank space is specified as a terminator (alternately, @ could have been used).

In this POSTPATH procedure, **PATH = 2** because sequential access is done via PATH 2, as designated by the default QDESIGN-generated **FIND** procedure (access via the IMAGE master search field is done through PATH 1). When **GET...SEQUENTIAL** is subsequently called for PATH 2 in the **FIND** procedure, SUPERDEX access is automatically performed instead.

If multiple databases are configured in the data dictionary, specify the name of the **SIPROC** file that references the corresponding database, for example:

```
GET SIPROC1 USING SIPARMS OPTIONAL
```

The primary file is specified with **OPEN_n** to force a separate access path. Use the **SIPROC_n** on the **GET** that matches the number on the **OPEN_n**.

 **If you are using version 6.09 and are having problems getting any records to qualify, add the NOGENERIC clause to the GET statement in the POSTPATH procedure.**

Qualifying entries for retrieval

In QUICK, entries in IMAGE datasets are generally retrieved either by key (using **GET...VIA**) or via a serial read (using **GET...SEQUENTIAL**). Fast access is normally attained only when specifying the full, exact key value for an entry.

Using SUPERDEX, QUICK is able to rapidly qualify multiple entries based on various criteria, and may be used to refine the selection using various fields, datasets, and databases.

Entries qualified in QUICK may be retrieved in ascending or descending sorted sequential order without the need to sort.

Differences between the QUICK interface and SUPERDEX's DBFIND intrinsic

Examples of various types of retrievals are documented in the *Section 4: Programming* section of the *SUPERDEX User Manual*.

The main difference between retrievals using the QUICK interface and SUPERDEX's DBFIND intrinsic is that the terminology is altered. The following table shows the equivalent QUICK interface terminology vs. DBFIND:

QUICK	DBFIND	description
SIDSET	<i>dset</i>	TurboIMAGE dataset name
SIMODE	<i>mode</i>	mode of operation (see below for discussion)
SIPATH	<i>item</i>	SUPERDEX SI-path (optionally followed by SI-LINK)
SIARG	<i>argument</i>	search value

In the QUICK interface, the **TRIM** function is useful for constructing complicated *SIARG* search values.

Special SIMODE values

To cause SUPERDEX'ed QUICK to return all the entries that are alphabetically greater-than the *SIARG* rather than just the entries that qualify based on the *SIARG* specified, add **1000** to or subtract **1000** from the *SIMODE* value.

By default, entries are returned by QUICK using SUPERDEX in ascending alphabetical order. To instead return entries in descending order, add **2000** to or subtract 2000 from the *SIMODE* value.

To cause QUICK with SUPERDEX to return all the entries that are alphabetically less-than the *SIARG*, add or subtract **3000** to or from the *SIMODE* value.

SIPARMS item

The parameters for SUPERDEX access are assigned values in the **POSTPATH** procedure, using QUICK standard **LET** statements.

Parameters

The name of the dataset to access is assigned in the *SIDSET* parameter, and the SI-path name is assigned in the *SIPATH* parameter. The *SIMODE* specifies how entries should be qualified and whether they should be returned in ascending or descending order. The search value (partial or generic key, range of values, or multiple search values) are specified with the desired operators in the *SIARG* parameter.

The parameters of *SIPARMS* have the following meaning:

SIDSET The **OPEN** name of the dataset in the QDESIGN file (which may differ from the **FILE** name used in the data dictionary). It must be enclosed in quotes and, if less than 16 characters, appended with a single space or semicolon.

SIPATH The name of the SI-path to access. It must be enclosed in quotes and, if less than 16 characters, appended with a single space or semicolon.

If performing successive **GET SIPROC...**s against multiple datasets, a common item used to logically link the files together (called the *SI-link*) may be additionally specified. The use of *SI-link* does have some requirements, and [Section 4](#) of the [SUPERDEX User Manual](#) should be consulted for questions.

The *SI-link* is separated from the SI-path name by a comma, with the combined value terminated by a single space or semicolon as shown:

```
LET SIPATH = "SI-path,SI-link;"
```

Additionally, a length parameter (called the *SI-link-length*) may be assigned to the SI-link value as follows (please refer to [Section 4: Programming](#) of the [SUPERDEX User Manual](#) for further information):

```
LET SIPATH = "SI-path,SI-link,SI-link-length;"
```

SIMODE The *SIMODE* is the same as the *mode* parameter used in SUPERDEX's DBFIND intrinsic (see [Section 5](#) of the [SUPERDEX User Manual](#)), but additional *modes* are also supported.

SIMODE 0 Resets to sequential access. The *SIPATH* and *SIARG* parameters are ignored and do not need to be specified.

SIMODE *n*+1000 By default, only entries that match the specified *SIARG* are returned. By adding 1000 to or subtracting 1000 from the *SIMODE*, all entries greater-than or equal-to the *SIARG* will be returned in ascending order.

SIMODE *n*+2000 By default, entries are returned in ascending sorted sequential order. By adding 2000 to or subtracting 2000 from the *SIMODE*, entries will instead be returned in descending order.

SIMODE *n*+3000 Same as *SIMODE* +1000, but returns entries less-than or equal-to the *SIARG* in descending order.

SIARG Value or values to search for. The *SIARG* value may contain all of standard capabilities, retrievals, and Boolean operators that are available in the standard SUPERDEX DBFIND.

SICOUNT item

In addition to the input parameters used in the *SIPARMS*, a separate item *SICOUNT* is used as an output parameter and returns the number of entries which were qualified by **GET SIPROC...** Not all *SIMODEs* return an *SICOUNT* value. For a list of modes which return a count of qualified record, see Section 5 of the *SUPERDEX User Manual*.

Sample QUICK screen

This example shows a simple QUICK program that accesses the manual master CUSTOMERS. If CUSTOMER-NUMBER (the TurboIMAGE key) is specified, then it will be used; if a value for another field is entered, the dataset is read sequentially.

```
SCREEN CUST  
  
FILE CUSTOMERS PRIMARY  
  
FIELD CUSTOMER-NUMBER  
FIELD CUSTOMER-NAME  
FIELD STATE  
FIELD PHONE-PREFIX
```

Below, the **FILE SIPROC DESIGNER ... ACCESS VIA SIPARMS** statement has been added, as well as a **POSTPATH** procedure to additionally facilitate SUPERDEX access using one of two SI-paths or to return the records using IMAGE access via a serial read:

```
SCREEN CUST

FILE CUSTOMERS PRIMARY
FILE SIPROC DESIGNER
  ACCESS VIA SIPARMS

FIELD CUSTOMER-NUMBER
FIELD CUSTOMER-NAME
FIELD STATE
FIELD PHONE-PREFIX

PROCEDURE POSTPATH
  BEGIN
    IF PATH = 2
      THEN BEGIN
        PROMPT CUSTOMER-NAME
        IF NOT PROMPTOK
          THEN LET PATH = 0
        ELSE BEGIN
          LET PATH = 2
          LET SIDSET = "CUSTOMERS;"
          LET SIPATH = "CUSTOMER-NAME;"
          LET SIMODE = 1
          LET SIARG = CUSTOMER-NAME + "@"
          GET SIPROC USING SIPARMS OPTIONAL
          IF NOT ACCESSOK
            THEN
              ERROR "No records with matching CUSTOMER-NAME."
          END
        END
      END

    IF PATH = 0
      THEN BEGIN
        PROMPT STATE
        IF NOT PROMPTOK
          THEN LET PATH = 0
        ELSE BEGIN
          LET PATH = 2
          LET SIDSET = "CUSTOMERS;"
          LET SIPATH = "STATE;"
          LET SIMODE = -102
          LET SIARG = STATE
          GET SIPROC USING SIPARMS OPTIONAL
          IF NOT ACCESSOK
            THEN
              ERROR "No records with matching STATE."
          END
        END
      END
    END
```

```
IF PATH = 0
  THEN BEGIN
    LET PATH = 2
    LET SIDSET = "CUSTOMERS;"
    LET SIMODE = 0
    GET SIPROC USING SIPARMS OPTIONAL
  END
END
```

In the example, SUPERDEX access is permitted by CUSTOMER-NAME and STATE using the SI-paths of the same names.

The screen works as follows:

1. CUSTOMER-NUMBER is accepted. If specified, the IMAGE path is used.
2. If CUSTOMER-NUMBER is not specified, CUSTOMER-NAME is accepted. If specified, the **SIPARMS** parameters are set to access the appropriate SI-path, and an @ is appended to the name specified to unconditionally perform a partial-key retrieval.
3. If CUSTOMER-NAME is not specified, STATE is accepted. The **SIPARMS** parameters establish the correct SI-path and set the *SIMODE* to -102, which specifies that only the first two bytes of the value are recognized.
4. If no value was entered for CUSTOMER-NUMBER, CUSTOMER-NAME, or STATE, a sequential search is performed. For this, the SI-path is reset (by specifying an *SIMODE* of 0), since it may have been set by a previous **GET**.

CHAPTER 4:

QUIZ interface

This chapter discusses SUPERDEX's interface to QUIZ. It is assumed that you have:

- loaded the tape according to the separate SUPERDEX loading instructions
- installed the PowerHouse interface as explained in the Installation chapter of the Configuration section

The first part in this chapter explains how to go about Invoking QUIZ to access SUPERDEX capabilities.

The next part describes the process of Modifying source files for SUPERDEX access in order to access entries using SUPERDEX.

The next part explains the differences in QUALIFYING entries for retrieval between the QUIZ interface and SUPERDEX's DBFIND intrinsic.

The next part overviews the new SIQUIZ Commands, as well as the impact of SIQUIZ on existing QUIZ commands.

The following three parts describe the new SIDEF command, used for defining and initializing SIQUIZ variables, the SIPATH command, used for defining the dataset, SI-Path, and prompt or argument; and other SIQUIZ commands which act like QUIZ commands.

The next part describes the impact on, considerations for, and other NOTES on existing QUIZ commands.

The last part shows before and after Sample source files modified for SUPERDEX access through SIQUIZ.

Invoking SIQUIZ

The **SIQUIZ** program is run as a front-end to QUIZ. This is done as follows:

```
:RUN SIQUIZ.SUPERDEX.SYS;INFO="parameter[parameter]..."
```

where parameter is one or more of those listed below, delimited by spaces. Strings other than the following are passed to QUIZ literally.

AUTO=*mpefile*

Equivalent to executing the statements:

```
USE mpefile  
GO  
EXIT
```

where *mpefile* is a QUIZ source or program file.

PROCLOC= *.group*[*.account*]

Specifies the group and account in which the QUIZ program resides, normally CURRENT.COGNOS.

Modifying source files for SUPERDEX access

The QUIZ interface uses the new **SIPATH** command to define the parameters for SUPERDEX access: the dataset, SI-path, and search argument. It is used in place of or in addition to the **SELECT** command.

For example, the following QUIZ report:

```
ACCESS CUSTOMERS
SELECT IF CUSTOMER-NAME = "ACME MANUFACTURING"
REPORT CUSTOMER-NUMBER CUSTOMER-NAME
GO
EXIT
```

would be modified for SUPERDEX's QUIZ interface as shown:

```
;SIPATH CUSTOMER-NAME "ACME MANUFACTURING "
ACCESS CUSTOMERS
REPORT CUSTOMER-NUMBER CUSTOMER-NAME
GO
EXIT
```

In the example above, the customer "ACME MANUFACTURING " (defined as the argument) is being searched for in the CUSTOMERS dataset via the SI-path CUSTOMER-NAME.

The parameters defined by the **SIPATH** command are as follows:

<i>dataset</i>	name of dataset containing entries (optional if same set as in ACCESS statement)
<i>SI-path</i>	name of SI-path to utilize
<i>argument</i>	search argument

The values for these parameters, along with the **SIPATH** command is documented fully in the *SIPATH command* chapter at the end of this section.

Additionally, the **SIDEF** command may be used like QUIZ's **DEFINE** command to define and initialize variables for use in **SIPATH** commands.

For example, the following QUIZ report:

```
ACCESS CUSTOMERS
DEFINE CUST-TEMP CHAR*30 = PARM &
  PROMPT "Enter customer name:"
SELECT IF CUSTOMER-NAME = CUST-TEMP
REPORT CUSTOMER-NUMBER CUSTOMER-NAME
GO
EXIT
```

would be modified for SUPERDEX's QUIZ interface as shown:

```
;SIDEF CUST-TEMP PROMPT "Enter customer name:"
;SIPATH CUSTOMERS.CUSTOMER-NAME CUST-TEMP "@"
ACCESS CUSTOMERS
REPORT CUSTOMER-NUMBER CUSTOMER-NAME
GO
EXIT
```

This example will accept the input from the user with the **SIDEF** command. It will then append the entered data with "@" using the **SIPATH** command. This will force a partial lookup, without the user needing to specify the "@".

Qualifying entries for retrieval

In QUIZ, entries in IMAGE datasets are generally retrieved either by key (using **CHOOSE**) or via a serial read (using **SELECT**). Fast access is normally attained only when using **CHOOSE**, for which the full, exactly-matching key value for an entry must be specified.

Using SUPERDEX, QUIZ is able to rapidly qualify multiple entries based on various criteria, and may be used to refine the selection using various fields, datasets, and databases.

Entries qualified in QUIZ may also be retrieved in ascending sorted sequential order without the need to sort.

Differences between the QUIZ interface and SUPERDEX's DBFIND intrinsic

Examples of various types of retrievals are documented in the *Programming* section of the **SUPERDEX User Manual**.

SIPATH Command	DBFIND	Description
DATASET	<i>Dset</i>	Dataset (Optional in QUIZ)
SIPATH	<i>Item</i>	SUPERDEX SIPATH (Optionally followed by SI-link)
PROMPT TEXT	None	Displayed text
ARGUMENT	<i>Argument</i>	Search value (QUIZ hardcoded)
Search Criteria	<i>Argument</i>	Search value (QUIZ reply from prompt)
None	<i>Mode</i>	Mode of operation (Mode 21 used in QUIZ)

Restrictions: Relational retrieval using multiple sets can be used to restrict the number of qualifying entries from the primary file, based on search criteria specified for **L**INKed files in the *same database*. The **L**INKed files are accessed by QUIZ through normal IMAGE access. It is therefore necessary to keep the **S**ELECT commands for **L**INKed files.

SIQUIZ Commands

Two new commands, which are similar to QUIZ commands, are provided in SIQUIZ to facilitate SUPERDEX access against SI-paths:

- SIDEF** similar to QUIZ's **DEFINE** command, but used only for SUPERDEX access on an SI-path
- SIPATH** similar to QUIZ's **SELECT** command, but used only for SUPERDEX access on an SI-path

Additional SIQUIZ commands work the same in SIQUIZ as they do in QUIZ:

- USE** specifies the QUIZ source file to execute
- GO** invokes the **USE**d QUIZ source file
- EXIT** ends and suspends the program
- QUIT** ends and terminates the program

The new **SIDEF** and **SIPATH** commands may affect the use of the following QUIZ commands:

- CHOOSE** may not be used to access SI-paths, only IMAGE paths (use **SIPATH** instead)
- DEFINE** may not be used for SI-paths, only IMAGE paths (use **SIDEF** instead)
- SELECT** may be eliminated or modified to perform access not done by SUPERDEX's **SIPATH** command
- SORT** may be eliminated, since entries are returned in sorted order

**SEARCH
CRITERIA**

The value or values to search for on the referenced SI-path, in response to the *prompt text*. This is similar to the *argument*, but some constructs are not supported and an easier method is provided for users to specify multiple values for relational access.

Refer to the discussion of search criteria in the *SIPATH command* chapter in this section for more information.

EXPRESSION

String constant to assign to the referenced variable. May also consist of multiple string constants, enclosed in quotes, and variables previously defined using the **SIDEF** command.

Can also contain the QUIZ reserve word **SYSDATE**, with an arithmetic expression (+ number of days or - number of days). The variable would contain a valid date in the YYMMDD ASCII format

When referencing a variable previously defined in a **SIDEF** command, substrings can be specified. For example:

```
;SIDEF TODAY SYSDATE  
;SIDEF YEARMONTH TODAY [1:4]  
;SIDEF MONTH TODAY [3:2]
```

The format for substring designation is [START POSITION:LENGTH]

The **SIPATH** parameters are defined as follows:

- DATASET** Optional parameter. Name of the dataset to access, specified only for retrievals against a dataset other than the primary dataset (the first one specified in the QUIZ **ACCESS** command).
- SI-PATH** The name of the SI-path to use for the retrieval.
- SI-LINK** Optional parameter. The name of an item that appears in multiple datasets which is used to logically link the sets, or to impose sorting order in relational access mode. The SI-link item must be configured as an SI-subkey in a concatenated SI-key or a search field for an SI-path related to a master dataset.
- PROMPT** This specifies that the user will supply the search criteria. If **Rprompt** is used, **Rprompt** relational access for the search criteria will be enforced.
- PROMPT TEXT** Issued as a user prompt with the response passed as the *search criteria*.
- ARGUMENT** String expression which may include constants and/or variables (created using the **SIDEF** command) concatenated together and delimited by a single space.
- The *argument* may contain many of the capabilities, retrievals, and Boolean operators supported by SUPERDEX. See the *SUPERDEX User Manual, Section 5*.
- SEARCH CRITERIA** The value or values to search for on the referenced SI-path, in response to the *prompt text*. This is similar to the *argument*, but some constructs are not supported and an easier method is provided for users to specify multiple values for relational access.

BOOLEAN

Boolean operations between multiple values may be specified using the following Boolean operators:

Boolean operators	
+	and
,	or
-	and not

The Boolean operators are specified as separators with no spaces and optionally delimited with parentheses, as shown:

(A@+B@) , (P@-PR@)

This locates all the entries in a keyworded SI-path with words that begin with "A" and "B", or that begin with "P" and not "PR".

In this example, the parentheses are not required because the **+** (AND) and **-** (NOT) have precedence over **,** (OR).

**MULTIPLE
SI-PATHS
RETRIEVAL**

For retrievals that require the use of multiple SI-path commands, the *search criteria* of all **except the first** SI-path command must be prefixed by a Boolean operator (**+**, **,**, or **-**) specifying how the retrieval by the current SI-path is to be related to the previous SI-path. Additionally, search criteria must be prompted for with RPROMPT, not PROMPT.

Restrictions: Two restrictions exist in the *search criteria* that may be specified:

- for range searches as well as searches that use values that start with the **<=**, **>=**, or **<>** relational operators, an **?** embedded in a value is not recognized as an operator but as a regular character
- values for alphanumeric SI-keys (data types U and X) must be appended with **@** when specified with the relational operators **<=**, **>=**, or **<>**

For complete restrictions, see Section 4 of the SUPERDEX User Manual.

Other SIQUIZ commands

USE command

Once the **SIPATH** command has been issued (either via the SIQUIZ program or embedded in the source program), SIQUIZ's **USE** command specifies which QUIZ source file should be executed:

```
>USE QUIZ-program
```

where **QUIZ-program** is the name of the QUIZ source file.

Lines in the QUIZ source file may **NOT** be more than 72 characters in length. Additionally, nested **USE** files may not contain SIQUIZ commands--all SIQUIZ commands must be contained in the first **USE** file.

GO command

After executing the USE command, the **GO** command invokes the referenced QUIZ source file:

```
>GO
```

This command has no parameters.

EXIT command

After the QUIZ program has executed, the **EXIT** command may be used to exit and suspend the SIQUIZ program:

```
>EXIT
```

This command may be abbreviated to **EXI**, **EX**, or **E**.

QUIT command

After the QUIZ program has executed, the **QUIT** command may be used to exit and terminate the SIQUIZ program:

```
>QUIT
```

Notes on QUIZ commands

CHOOSE command

QUIZ's **CHOOSE** command works only on IMAGE paths and may not be used to access SUPERDEX SI-paths--the **SIPATH** command must be used instead.

DEFINE command

QUIZ's **DEFINE** command works only on IMAGE paths and may not be used to set up definitions for SUPERDEX SI-paths--the **SIDDEF** command must be used instead.

SELECT command

The **SELECT** command is either replaced by or modified for SIQUIZ's **SIPATH** command.

If the **SIPATH** command is functionally equivalent to the **SELECT** command, the **SELECT** command may be removed and replaced by the **SIPATH** command; otherwise, if the **SIPATH** command is added and the **SELECT** command is not removed it is functionally ignored.

If the required access cannot be accomplished by one or more **SIPATH** commands, the **SELECT** command should be modified to perform only the additional access needed.

If the required access is against a non-primary (**LINK TO**) file, the **SELECT** command should be modified to perform this access. An **SIPATH** command against a non-primary file simply de-selects primary entries which do not have at least one qualifying secondary entry.

SORT command

Because entries on an SI-path which are qualified using the **SIPATH** command are automatically returned in ascending sorted order, any corresponding **SORT** command may be eliminated from or replaced by a **SORTED** command in the QUIZ source file. You may continue to use the **SORT** command to print or save the records in descending order.

Sample source files

This example shows an unmodified QUIZ source file that accesses the manual master EMPLOYEES using the field LAST-NAME, which is not a key:

```
ACCESS EMPLOYEES
DEFINE EMPLOYEE-LAST CHARACTER*30 = PARM &
  PROMPT "Enter last name:"
SELECT IF LAST-NAME = EMPLOYEE-LAST
REPORT FIRST-NAME LAST-NAME STATE
GO
EXIT
```

This report is modified for SUPERDEX access in SIQUIZ by replacing the **SELECT** command with an **SIPATH** command that accesses the SI-path named LAST-NAME-KEY, prefixed with a semicolon (;), as the first command in the source file:


```
;SIPATH LAST-NAME-KEY PROMPT "Enter last name:"
ACCESS EMPLOYEES
REPORT FIRST-NAME LAST-NAME STATE
GO
EXIT
```

This example is of a QUIZ program that additionally accesses the HOURS-WORKED dataset:

```
ACCESS EMPLOYEES LINK TO HOURS-WORKED
DEFINE EMPLOYEE-LAST CHARACTER*30 = PARM &
  PROMPT "Enter last name:"
DEFINE WEEK-ENDING DATE = "Enter end day of week:"
SELECT IF LAST-NAME = EMPLOYEE-LAST AND DATE = WEEK-ENDING
REPORT FIRST-NAME LAST-NAME STATE
GO
EXIT
```

Modifying for SIQUIZ requires two **SIPATH** commands and a change to the **ACCESS** command. Now, only the primary file (*EMPLOYEE*) is specified by the **ACCESS** command, with the other dataset included in the first **SIPATH** command:

```
;SIPATH EMPLOYEES.LAST-NAME-KEY R;SIPATH HOURS-WORKED.WEEK-ENDING RACCESS EMPLOYEES  
REPORT FIRST-NAME LAST-NAME STATE  
GO  
EXIT
```

 **The user must append an + to the search argument for the second prompt to inform SUPERDEX that an AND operation is to be performed with the first prompt.**

CHAPTER 5:

QTP interface

This section discusses SUPERDEX's interface to QTP. It is assumed that you have:

- loaded the tape according to the separate SUPERDEX loading instructions
- installed the PowerHouse interface as explained in the Installation chapter of the Configuration section

The only part in this chapter describes how to go about Invoking SIQTP.

Because the QTP interface operates identically to the QUIZ interface, refer to the QUIZ Interface chapter for details.

Invoking SIQTP

The **SIQTP** program is run as a front-end to QTP. This is done as follows:

```
:RUN SIQTP.PUB.SUPERDEX;INFO="parameter[parameter]..."
```

where parameter is one or more of the strings defined in the QUIZ interface section, or passed literally to QTP.

Operating SIQTP

The SIQTP program is used in the same way as SIQUIZ, as explained in the previous section.

CHAPTER 6:**Error and exceptional conditions****SIQUIZ and SIQTP errors and warnings**

The **SIQUIZ and SIQTP errors and warnings** table lists the various error messages that could be issued by the SIQUIZ or SIQTP programs, their meanings, and their corrective actions.

SIQUIZ and SIQTP errors and warnings

Messages prefixed with ***E*** are designated as **errors**, while those prefixed with ***W*** are **warnings**.

Type	Description
Message Meaning Action	*W* COMPILED REPORT, PLEASE ENTER SOURCE FILE A compiled report was USEed and no SIPATH statement was specified in SIQUIZ/SIQTP. Because SIPATH statements embedded in source files are dropped when they are compiled, either specify an SIPATH statement in SIQUIZ/SIQTP or USE a source file rather than a compiled report.
Message Meaning Action	*W* COMPILED REPORT, UNABLE TO SPECIFY SIPATH A compiled report file was specified with the AUTO option. Specify a source file instead.
Message Meaning Action	*E* FILE XXXXXXXX NOT FOUND The specified source file does not exist. Make sure entered correctly. Qualify with group and account if necessary.
Message Meaning Action	*E* MISSING VARIABLE An SIPATH command references a variable that was not already initialized by an SIDEF command. Variables must first be initialized by the SIDEF command.
Message Meaning Action	*W* NO PATH DEFINED The source file does not contain an SIPATH statement, and no SIPATH statement was issued in SIQUIZ/SIQTP. First specify an SIPATH statement either in the source file or in SIQUIZ/SIQTP.
Message Meaning Action	*E* NO REPORT TO RUN GO was entered without a previous USE statement. First specify the file to USE .
Message Meaning Action	*W* NO SIPATH STATEMENT FOUND IN XXXXXXXX The source file USE d does not contain an SIPATH statement, and no SIPATH statement was issued in SIQUIZ/SIQTP. If the correct file name was specified, enter the appropriate SIPATH statement in SIQUIZ/SIQTP and GO .
Message Meaning Action	*E* QTP CANNOT BE RUN SIQTP cannot create the QTP process, typically because PROCLOC= has not been specified. Correct the condition that caused the error and rerun SIQTP.

SIQUIZ and SIQTP errors and warnings (cont'd)

Type	Description
<i>Message</i> <i>Meaning</i> <i>Action</i>	*E* QUIZ CANNOT BE RUN SIQUIZ cannot create the QUIZ process, typically because PROCLOC= has not been specified. Correct the condition that caused the error and rerun SIQUIZ.
<i>Message</i> <i>Meaning</i> <i>Action</i>	*E* TOO MANY SIDEFS More than 16 SIDEF statements have been entered. Do not exceed the maximum limit.
<i>Message</i> <i>Meaning</i> <i>Action</i>	*E* UNKNOWN STATEMENT An unrecognized statement has been specified. Specify only valid SIQUIZ/SIQTP statements; check your spelling.

SECTION 3:

Speedware

This section discusses the interface to Speedware Corp.'s Speedware product.

 **This interface only works with SPEEDWARE Version 6.**

Chapter 1 Function	Overview Identifies the basic concepts of the interface and the SUPERDEX access types.
Chapter 2 Function	Installation Describes the steps necessary to execute the interface.
Chapter 3 Function	Designer Interface Gives a detailed layout of the interface modifications to a Designer screen.

CHAPTER 1:

Overview

Introduction

The SUPERDEX/Speedware interface utilizes SUPERDEX's standard Fourth-Generation Language interface. This chapter explains how the Speedware interface works, along with an overview of various Access Methods available through SUPERDEX's Fourth-Generation Language Interface and Speedware.

Qualifying and retrieving entries

The Speedware interface works by reinterpreting the sequential access defined with the **SELECT** verb to perform sorted indexed-sequential SUPERDEX access rather than an IMAGE serial read. To accomplish this, only minor additions to Designer screens are required, and no changes to default statements are required.

In each Designer screen, statements to set up parameters for SUPERDEX access are written using the standard **LET** command. The dataset to access is defined in the *SIDSET* parameter; the SI-path is defined in the *SIPATH* parameter; and the *SIMODE* parameter specifies how entries should be qualified and whether they should be returned in ascending or descending order. The search value, partial or generic key, range of values, or multiple search values are specified with the desired operators in the *SIARG* parameter.

Because most of SUPERDEX's capabilities are accomplished by embedding multiple values and conditional, relational, and Boolean operators in the search argument, it is normally sufficient to simply vary the value of the *SIARG* and use an *SIMODE* of 1.

Entries are by default returned in ascending sorted sequential order, and may alternately be returned in descending order by altering the *SIMODE*. Other *SIMODE*s may be used to additionally retrieve entries that are alphabetically higher or lower than the *SIARG* value and to perform other specialized functions, such as returning the alphabetically highest or lowest entry in the dataset.

SUPERDEX automatically redefines the access performed by the **SELECT** verb to be an indexed-sequential read via SUPERDEX rather than an IMAGE serial read. After entries are retrieved, the qualifying number of entries are returned in the **G-SICOUNT** item when *SIMODE* 1 is used.

Access methods

Multiple keys in master and detail datasets

The **LET** commands are written to set up the parameters for SUPERDEX access, including the SI-path to access, and **DBGET** is called. When the retrieval is done via the **SELECT ***, an indexed-sequential read is performed by SUPERDEX rather than a serial read.

Concatenated keys containing multiple fields

The *SIARG* item is redefined as necessary to represent the SI-subkeys in the concatenated SI-key, their lengths, and their data types. The values for each SI-subkey are defined as separate **LET** commands.

Sorted sequential retrieval

Entries are unconditionally returned in sorted sequential order for entries qualified in indexed access mode. In relational access mode, an SI-link may be specified in the *SIPATH* parameter to enforce sorting order.

By default, entries that match the specified *SIARG* are returned in ascending order. By adding 2000 to the *SIMODE*, entries are instead returned in descending order. All the entries in a dataset may be read in ascending or descending sorted order by using *SIMODE* 1100 or 3100, respectively.

Keyword retrieval



Keyword retrieval is available only in the SUPERDEX II package.

Keyworded SI-paths may be accessed through the screens without restriction.

Generic and partial key retrieval

Partial key access can be performed in SUPERDEX'ed Speedware using two different methods.

The first is to use an *SIMODE* of 1 and specify the partial key value appended with an **@** as the *SIARG* (e.g. **HEWL@**). This will locate all entries that match on the significant characters followed by anything.

The second method is to specify the value in the *SIARG* without an @ but vary the *SIMODE* based on the length of the value in the *SIARG*. For example, an *SIARG* containing the partial key **ROLA** would dictate *SIMODE* 102 or -104 (100 plus the number of words or bytes, respectively, in the value).

Generic key retrieval is accomplished by embedding the ? matchcode in the *SIARG*, which holds the place of any alphanumeric character. For example, the *SIARG* **L?TTER** would locate "LETTER" and "LITTER"; by appending an @ (**L?TTER@**), "LETTERMAN", "LITTERBUG", and "LOTTERY" would also be located.

Approximate match retrieval

Approximate match retrieval is performed by using an *SIMODE* that specifies how many characters in the *SIARG* SUPERDEX should match on, which is typically the length of the value specified. If no matching entry exists, the nearest matching entry is returned.

The *SIMODE* also dictates whether the internal SI-pointer in the B-tree should be set before or after the matching or nearest matching entry, and whether entries should be returned in ascending or descending order.

For example, an *SIARG* containing the value **UNITED** would dictate *SIMODE* 1103 or -1106, both of which would cause SUPERDEX to match on the entire value. The *SIMODE* is calculated as 100 plus the number of words or bytes (negated if bytes), plus or minus 1000 to dictate ascending order retrieval. Using these *SIMODE*s, the SI-pointer would be set before the matching or nearest matching entry. Any entry beginning with "UNITED" would be included in the retrieval.

If *SIMODE* 3103 were instead specified (adding 3000 instead of 1000 for descending order), any "UNITED" entry would be excluded, since the SI-pointer is set before the matching entry and entries are being returned in descending order. With an *SIMODE* of 3203 or -3206 (200 plus the number of words or bytes, plus or minus 3000 for descending order retrieval), the SI-pointer would be set after the matching or nearest matching entry, and the "UNITED" entries would now be included.

Greater-than or equal-to, less-than or equal-to, and range retrieval

These retrievals are performed by embedding special operators in the *SIARG* for *SIMODE* 1 or 21.

Greater-than-or-equal-to retrieval is accomplished by prefixing the search value with the **>=** operator (e.g. **>=1000**), less-than-or-equal-to retrieval uses the **<=** operator as a prefix, and not-equal-to retrieval uses the **<>** operator. The **<>** operator can also appear after another value in the same *SIARG* to perform a Boolean AND NOT operation (e.g. **SUPER@<>SUPERDEX**).

Range retrievals are performed by using the **>=** and **<=** operators in combination. For example, a range search to find all entries with amounts between 500 and 1000, inclusive, is specified with the *SIARG* **>=500<=1000**.

By default, entries are returned in ascending sorted order, but may be returned in descending order by adding 1000 to the *SIMODE*.

Grouped retrieval

 **Grouped retrieval is available only in the SUPERDEX II package.**

Whenever the group is referenced by its SI-path name in the *SIPATH* parameter of *SIPROC*, all SI-keys that form the group are unconditionally searched.

Grouped SI-paths are accessed in the same way as non-grouped SI-paths -- the only difference is in the configuration of the SI-path.

There may be some ambiguity in searching by an SI-key in a grouped SI-path whose item length is shorter than the group length and which is therefore padded with spaces. For example, if CITY, an X16, and STATE, an X2, are grouped together with an SI-key length of 8 words (to accommodate CITY), an *SIARG* of **CA** would find not only all entries in the state of "California" but also those in the cities of "CALABASAS" and "CARLSBAD". To resolve this ambiguity, use *SIMODE* 1 or 21 and pad the *SIARG* with enough trailing spaces to cover the full SI-key length.

Super-grouped retrieval

 **Super-grouped retrieval is available only in the SUPERDEX II package.**

To reference a super-grouped SI-path, use the IMAGE master set as the set name on the **SELECT** verb. Speedware will search all paths related to the super-grouped SI-path in the master and detail datasets, but will only display the master entries.

Relational access: multiple criteria retrieval

 **Relational access with multiple criteria is available only in the SUPERDEX II package.**

Boolean operations are accomplished by embedding the appropriate operators in the search value in the *SIARG*.

For example, a search for all part descriptions that contain both the words PAPER and CLIP would be specified with an *SIARG* of **~PAPER AND CLIP;** or **~PAPER+CLIP;**. To find all invoices that are unpaid or canceled, the *SIARG* would be **~UNPD OR CANC;** or **~UNPD,CANC;**; and to find all entries in California and not Los Angeles, **~CA NOT "LOS ANGELES";** or **~CA-"LOS ANGELES";**.

Relational access: multiple fields, sets, bases



Relational access with multiple fields, sets, and bases is available only in the SUPERDEX II package.

Multiple **DBGET** calls are performed in succession for each SI-path. If performing retrievals against multiple fields or sets within a database, the *SIDSET* and *SIPATH* parameters are varied to specify the desired access. If performing retrievals against multiple bases, the **\$BASE** must specify the correct base.

Custom indexing

The custom SI-path is referenced in the *SIPATH* parameter and the custom SI-key value is specified in the *SIARG* parameter.

CHAPTER 2:

Installation

Installation

Once the SUPERDEX installation tape has been loaded, as described in the separate SUPERDEX loading instructions, only minor modifications are necessary to the data dictionary. Refer to the Configuration/Establishing SI-indices section of the SUPERDEX User Manual.

Entry Maintenance

Speedware is automatically executed through the SUPERDEX XL ([XLSUPRDX.PUB.SYS](#)) when the database is enabled for SUPERDEX, so user records that are added, updated, or deleted from within Speedware will also maintain the SI-indices automatically.

Data Dictionary changes for Designer

The following changes should be applied to your current Speedware Data Dictionary. The changes include adding a dummy Detail dataset named SI for each database that will be accessed by SUPERDEX. Additionally, some Global (**No Share**) variables need to be defined.



At least one index needs to have been previously established for the current database so that the SI-dataset and SI-item will have been established. This is necessary for compiling the data dictionary.

Data Dictionary

1. Use DBUTIL to disable indexing on the database.

```
:RUN DBUTIL.PUB.SYS
>>DISABLE <base> FOR INDEXING
    Indexing is disabled for SUPERDEX

    Indexing is disabled
>>EXIT
```

2. Run Query to identify the actual size and structure of the SI item (to be used in step 6 later). For example:

```
:RUN QUERY.PUB.SYS
>B=<your base name>
PASSWORD = >> (your database password)
MODE = >> 5
> FORM SI

SET NAME:
    SI,DETAIL

    ITEMS:
        SI,                4X254

    CAPACITY: 4000          ENTRIES: 713
```

3. Obtain a current copy of your Data Dictionary.

4. Proceed to the Speedware/Designer Main menu and select option 1. Data Dictionary.
 1. Data Dictionary
5. From the Data Dictionary Menu, select option 5. File Definition.
 5. File Definition
6. On the File Definition Menu, select function key **F4** for New File. Enter the file name SI with one item also named SI. The Database name is the name of the Database you are applying SUPERDEX to. The format for the item is the same as displayed via Query in step two. Since you need to access a dataset that is not in the screen/report, for ease of use, it is also recommended that you add a second file named DUMMY with one item DUMMY-ITEM. Repeat this step for each Database.

Global Variables

For Designer screens to qualify and retrieve records using SUPERDEX, Speedware Global Variables must be set up. These Global variables must be defined as **No Share**.

<i>Name</i>	<i>Parm</i>	<i>Type</i>	<i>Size</i>	<i>Occurs</i>	<i>Storage</i>	<i>Bytes</i>	<i>Values</i>
G-SIPROC		Alpha	16.				SIPROC
G-SIIMAGEMODE		Numeric	4.		Int	2	7
G-SISTATUS		Alpha	20.				
***G-SICONDWORD		Numeric	4.		Int	2	
***G-SIEXTRA1		Alpha	18.				
G-SILIST		Alpha	2.				@;
G-SIBUFFER		Alpha	204.				
*G-SIPARMS		Alpha	200.				
**G-SIDSET		Alpha	16.				
**G-SIPATH		Alpha	34.				
**G-SIMODE		Numeric	4.		Int	2	
**G-SIARG		Alpha	148.				
*G-SICOUNT		Numeric	9.		Int	4	

* The variables G-SIPARMS and G-SICOUNT are used to parse the G-SIBUFFER variable.


**These variables are used to parse the G-SIPARMS variable.

***These variables are used to parse the G-SISTATUS variable.

The SIARG variable may be further parsed to support concatenated SI-keys that are of multiple data types. For example:

<i>Name</i>	<i>Parm</i>	<i>Type</i>	<i>Size</i>	<i>Occurs</i>	<i>Storage</i>	<i>Bytes</i>	<i>Values</i>
**G-SIARG		Alpha	148.				
***G-SIARG1		Numeric	9.		Int	4	
***G-SIARG2		Alpha	144.				

*** These two variables parse the G-SIARG variable.


 These Global variables can be alternatively defined as Local Variables. The disadvantage of using local variables is that you will need to define all of these variables in each screen and/or report or alternatively you can declare them in a Logic section and include them in the AT START section.

7. After compiling the dictionary with the **GO** command, the database must be enabled for indexing with SIMAINT.SUPERDEX.SYS.

```

:RUN SIMAINT.SUPERDEX.SYS
SIMAINT Version 4.2 (0f0a5c.FF) Copyright Dr. Matt / IABG (1988,1993)

Database >OEDB
Dataset > RETURN
    
```

 If Speedware requires recompilation of the catalog (e.g. because changes were made to the database) , you will have to repeat steps 1 AND 7 to Disable the database for indexing, do the GO and then re-enable the database for indexing.


CHAPTER 3:**Designer Interface**

The Designer interface reinterprets the **SELECT *** command to perform sorted indexed-sequential SUPERDEX access instead of an IMAGE serial read. Only minor additions to Designer screens are required to accomplish this. No changes to the default statements are required.

Screen Modifications

The SUPERDEX Speedware Interface reinterprets the sequential access from the **SELECT** verb to perform SUPERDEX retrievals. Only minor additions to the Speedware screens are required to accomplish this. These additions do not affect the default statements and processing of DESIGNER.

For each field within the **Select Level** of the **Logic Section** that will utilize SUPERDEX lookups and retrievals, a few additions are made. For example:

 The complete code has been included in each of the following examples for ease of readability. Since the following code is contained in every program (a second version without the printing of the number of records returned could be used in programs that don't use an SIMODE that returns the count) , it could be placed in a LOGIC program and then a #INCLUDE could be substituted in the code.

```
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
CALL DBGET USING $BASE(DUMMY),
               G-SIPROC,
               G-SIIMAGEMODE,
               G-SISTATUS MODIFY,
               G-SILIST,
               G-SIBUFFER,
               G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
  PRINT "CONDWORD = " + $STR(G-SICONDWORD)
  CURSOR (19,15) HILITE (ENH=HALF-INV);
IF G-SICOUNT = 0 THEN
  PRINT "NO MATCHING RECORDS"
  CURSOR (19,15) HILITE (ENH=HALF-INV);
ELSE
  PRINT $STR(G-SICOUNT) + " RECORDS FOUND"
  CURSOR (19,15) HILITE (ENH=HALF-INV);
```

Simple Key

```

SCREEN-SIMPLE-KEY:
AT SELECT DO BEGIN
(*
    This code runs the select screen when this program is
    started.
    The user enters either a partial Customer Name and a
    direction.
    The value entered is returned to this screen, where the
    appropriate record(s) are selected.
*)
SCREEN-CUSTOMER-SELECT USING L-CUSTOMER-NAME MODIFY,
                           L-DIRECTION      MODIFY;
(*
    Note: Refer to the Glossary for definitions of the Global
    Variables
    If a keyed selection is entered in the Customer Name, this
    code
    will select the records matching the partial Customer Name.
*)
G-SIARG  = $TRIM(L-CUSTOMER-NAME) + ";";
G-SIDSET = "CUSTOMERS;";
G-SIPATH = "CUSTOMER-NAME;";
IF L-DIRECTION = "B" THEN
    G-SIMODE = 2001;
ELSE
    G-SIMODE = 1;
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
CALL DBGET USING $BASE(DUMMY),
                G-SIPROC,
                G-SIIMAGEMODE,
                G-SISTATUS MODIFY,
                G-SILIST,
                G-SIBUFFER,
                G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
          CURSOR (19,15) HILITE (ENH=HALF-INV);
IF G-SICOUNT = 0 THEN
    PRINT "NO MATCHING RECORDS"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
ELSE
    PRINT $STR(G-SICOUNT) + " RECORDS FOUND"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
SELECT * FROM CUSTOMERS;
END;

```

Concatenated keys containing multiple fields

```

SCREEN-CONCATENATED:
AT SELECT DO BEGIN
(*
    This code runs the screen where the user can enter the
    keyed selection data.
*)
SCREEN-ORD-PART-SELECT USING L-ORDER-NUMBER  MODIFY,
                          L-PART-NUMBER  MODIFY;

G-SIARG1 = L-ORDER-NUMBER;
G-SIARG2 = L-PART-NUMBER;
G-SIARG  = $STR(G-SIARG1[1:4]) + G-SIARG2;
G-SIDSET = "ORDER-LINES;";
G-SIPATH = "ORDER-PART;";
G-SIMODE = -104 - $LEN($TRIM(L-PART-NUMBER));
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
CALL DBGET USING $BASE(DUMMY),
                G-SIPROC,
                G-SIIMAGEMODE,
                G-SISTATUS MODIFY,
                G-SILIST,
                G-SIBUFFER,
                G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
          CURSOR (19,15) HILITE (ENH=HALF-INV);
IF G-SICOUNT = 0 THEN
    PRINT "NO MATCHING RECORDS"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
ELSE
    PRINT $STR(G-SICOUNT) + " RECORDS FOUND"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
SELECT * FROM ORDER-LINES;
END;

```


Keyword retrieval



Keyword retrieval is available only in the SUPERDEX II package.

```

SCREEN-KEYWORDED-KEY:
AT SELECT DO BEGIN
  (*
    This code runs the select screen when this program is started.
    The user enters a partial Customer Name.
    The value entered is returned to this screen, where the
    appropriate record(s) are selected.
  *)
SCREEN-KEYWORD-SELECT USING L-CUSTOMER-NAME MODIFY;
  (*
    Note : Refer to the glossary for definitions of the Global Variables
    If a keyed selection is entered in the Customer Name, this code
    will select the records matching the partial Customer Name.
  *)

  IF L-CUSTOMER-NAME <> " " THEN
    BEGIN
      G-SIARG  = L-CUSTOMER-NAME + ";";
      G-SIDSET = "CUSTOMERS;";
      G-SIPATH = "CUSTOMER-NAME-KW;";
      G-SIMODE = 12;
      G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-
SIARG;
      G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
      CALL DBGET USING $BASE(DUMMY),
        G-SIPROC,
        G-SIIMAGEMODE,
        G-SISTATUS MODIFY,
        G-SILIST,
        G-SIBUFFER,
        G-SIPARMS MODIFY;
      G-SICONDWORD[1:2] = G-SISTATUS[1:2];
      G-SICOUNT[1:4] = G-SISTATUS[9:12];
      IF G-SICONDWORD <> 0 THEN
        PRINT "CONDWORD = " + $STR(G-SICONDWORD)
          CURSOR (19,15) HILITE (ENH=HALF-INV);
      IF G-SICOUNT = 0 THEN
        PRINT "NO MATCHING RECORDS"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
      ELSE
        PRINT $STR(G-SICOUNT) + " RECORDS FOUND"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
      SELECT * FROM CUSTOMERS;
    END;
  END;
END;

```

Grouped retrieval




Grouped retrieval is available only in the SUPERDEX II package.

```

SCREEN-GROUPED-KEY:
AT SELECT DO BEGIN
  (*
    If a keyed selection is entered in the Address, this code will
    select the records matching the partial Address.
  *)
SCREEN-ADDRESS-SELECT USING L-ADDRESS MODIFY;

  G-SIARG  = $TRIM(L-ADDRESS) + ";";
  G-SIDSET = "CUSTOMERS;";
  G-SIPATH = "ADDRESS1-CITY-KW;";
  G-SIMODE = 12;
  G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
  G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
  CALL DBGET USING $BASE(DUMMY),
           G-SIPROC,
           G-SIIMAGEMODE,
           G-SISTATUS MODIFY,
           G-SILIST,
           G-SIBUFFER,
           G-SIPARMS MODIFY;
  G-SICONDWORD[1:2] = G-SISTATUS[1:2];
  G-SICOUNT[1:4] = G-SISTATUS[9:12];
  IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
          CURSOR (19,15) HILITE (ENH=HALF-INV);
  IF G-SICOUNT = 0 THEN
    PRINT "NO MATCHING RECORDS"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
  ELSE
    PRINT $STR(G-SICOUNT) + " RECORDS FOUND"
          CURSOR (19,15) HILITE (ENH=HALF-INV);
  SELECT * FROM CUSTOMERS;
END;
```

Relational access: multiple fields, sets, bases

 Relational access with multiple fields, sets, and bases is available only in the SUPERDEX II package.

```

SCREEN-RELATIONAL:
AT SELECT DO BEGIN

(*
    This code runs the select screen when this program is started.
    The user enters either a Customer Number, partial Customer Name
    or partial Address. The value entered is returned to this screen,
    where the appropriate record(s) are selected.
*)
SCREEN-ORDER-SELECT USING L-CUSTOMER-NAME MODIFY,
                        L-PART-NUMBER  MODIFY;

(*
    Note : Refer to the Glossary for definitions of the Global Variables
    The following code will select the records for the keyworded
    selection(s) entered on the select screen.
*)
G-SIARG  = '' + $TRIM(L-CUSTOMER-NAME) + '';
G-SIDSET = "CUSTOMERS;";
G-SIPATH = "CUSTOMER-NAME;";
G-SIMODE = 12;
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
CALL DBGET USING $BASE(DUMMY),
                G-SIPROC,
                G-SIIMAGEMODE,
                G-SISTATUS MODIFY,
                G-SILIST,
                G-SIBUFFER,
                G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
          CURSOR (19,15) HILITE (ENH=HALF-INV);
SELECT * FROM CUSTOMERS;
G-SIARG  = "[*] ";
G-SIDSET = "ORDER-HEADERS;";
G-SIPATH = "CUSTOMER-NUMBER;";
G-SIMODE = 1;
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];

```

```

CALL DBGET USING $BASE(DUMMY),
                G-SIPROC,
                G-SIIMAGEMODE,
                G-SISTATUS MODIFY,
                G-SILIST,
                G-SIBUFFER,
                G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
        CURSOR (19,15) HILITE (ENH=HALF-INV);
SELECT * FROM ORDER-HEADERS;
G-SIARG = "and " + $TRIM(L-PART-NUMBER) + ";";
G-SIDSET = "ORDER-LINES;";
G-SIPATH = "PART-ORDER;";
G-SIMODE = 12;
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
CALL DBGET USING $BASE(DUMMY),
                G-SIPROC,
                G-SIIMAGEMODE,
                G-SISTATUS MODIFY,
                G-SILIST,
                G-SIBUFFER,
                G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
        CURSOR (19,15) HILITE (ENH=HALF-INV);
IF G-SICOUNT = 0 THEN
    PRINT "NO MATCHING RECORDS"
        CURSOR (19,15) HILITE (ENH=HALF-INV);
ELSE
    PRINT $STR(G-SICOUNT) + " RECORDS FOUND"
        CURSOR (19,15) HILITE (ENH=HALF-INV);
SELECT * FROM ORDER-LINES;
END;

```

SuperSELECT access:

 **SUPERSELECT** access is available as an option to SUPERDEX.

```
SCREEN-SUPER-SELECT:
AT ACCEPT DO BEGIN
( *
  The following code uses Bradmark's SUPER SELECT to access the indexed
  data. Refer to the particular access methods if you need information
  about how to enter the appropriate selection criteria.
*)
ON TERMINAL
BEGIN
  COMMAND "FILE SSSCREEN=SSWINDOW";
  COMMAND "RUN SUPERSEL.SUPERDEX.SYS";
  COMMAND "RESET SSSCREEN";
  SELECT * FROM CUSTOMERS;
END;
REPORT-SUPER-SELECT-RPT;
END;
EXIT;
```

Simple Report

```

REPORT-SIMPLE-REPORT:
DATA EXTRACT LOGIC - AT START
(*
    This code runs the select screen when this program is started.
    The user enters either a Customer Number, partial Customer Name
    or partial Address.  The value entered is returned to this screen,
    where the appropriate record(s) are selected.
*)
SCREEN-CUST-ADDR-SELECT USING L-CUSTOMER-NO    MODIFY,
                             L-CUSTOMER-NAME  MODIFY,
                             L-ADDRESS        MODIFY;
(*
    If a Customer Number is entered on the select screen, this code will
    select the record of the Customer Number entered.
*)
IF L-CUSTOMER-NO <> 0 THEN
    SELECT * FROM CUSTOMERS WHERE CUSTOMER-NUMBER = L-CUSTOMER-NO;
(*
    Note: Refer to the Glossary for definitions of the Global Variables
    If a keyed selection is entered in the Customer Name, this code
    will select the records matching the partial Customer Name.
    If a keyed selection is entered in the Address, this code will
    select the records matching the partial Address.
*)
IF L-CUSTOMER-NAME <> " " THEN
    G-SIARG = $TRIM(L-CUSTOMER-NAME) + ";",
    G-SIPATH = "CUSTOMER-NAME-KW;";
ELSE
    G-SIARG = $TRIM(L-ADDRESS) + ";",
    G-SIPATH = "ADDRESS1-CITY-KW;";

G-SIDSET = "CUSTOMERS;";
G-SIMODE = 12;
G-SIPARMS = G-SIDSET + G-SIPATH + G-SIMODE[1:2] + G-SIARG;
G-SIBUFFER = G-SIPARMS + G-SICOUNT[1:4];
CALL DBGET USING $BASE(DUMMY),
                G-SIPROC,
                G-SIIMAGEMODE,
                G-SISTATUS MODIFY,
                G-SILIST,
                G-SIBUFFER,
                G-SIPARMS MODIFY;
G-SICONDWORD[1:2] = G-SISTATUS[1:2];
G-SICOUNT[1:4] = G-SISTATUS[9:12];
IF G-SICONDWORD <> 0 THEN
    PRINT "CONDWORD = " + $STR(G-SICONDWORD)
        CURSOR (19,15) HILITE (ENH=HALF-INV);
IF G-SICOUNT = 0 THEN
    PRINT "NO MATCHING RECORDS"

```

The parameters for the **DBGET** in all of the above examples are pre-defined, except for the *SIPARMS*. The *SIPARMS* are defined as follows:

<i>SIDSET</i>	name of dataset containing the entries
<i>SIPATH</i>	name of the SI-path to utilize
<i>SIMODE</i>	defines method of SUPERDEX access
<i>SIARG</i>	search argument

The values for *SIPARMS* are documented fully later in the chapter.

Qualifying entries for retrieval

In Speedware, entries in IMAGE datasets are generally retrieved either by key or via a serial read. Fast access is normally attained only when specifying the full, exactly matching key value for an entry.

Using SUPERDEX, Speedware is able to rapidly qualify multiple entries based on various criteria, and may be used to refine the selection using various fields, datasets, and databases.

Entries qualified in Speedware may be retrieved in ascending or descending sorted sequential order without the need to sort.

Differences between the Speedware interface and SUPERDEX's DBFIND intrinsic

Examples of various types of retrievals are documented in the *Programming* section of the **SUPERDEX User Manual**.

The main difference between retrievals using the Speedware interface and SUPERDEX's DBFIND intrinsic is that the terminology is altered. The following table shows the equivalent Speedware interface terminology versus DBFIND:

<i>Speedware</i>	<i>DBFIND</i>	<i>description</i>
<i>SIDSET</i>	<i>dset</i>	IMAGE dataset
<i>SIMODE</i>	<i>mode</i>	SUPERDEX mode (see following)
<i>SIPATH</i>	<i>item</i>	SUPERDEX SI-Path (optionally followed by <i>SI-LINK</i>)
<i>SIARG</i>	<i>argument</i>	Search Value

Special SIMODE values

To cause SUPERDEX'ed Speedware to return all the entries that are alphabetically greater-than the *SIARG* rather than just the entries that qualify based on the *SIARG* specified, add **1000** to or subtract **1000** from the *SIMODE* value.

By default, entries are returned by SUPERDEX'ed Speedware in ascending alphabetical order. To instead return entries in descending order, add **2000** to or subtract **2000** from the *SIMODE* value.

To cause SUPERDEX'ed Speedware to return all the entries that are alphabetically less-than the *SIARG*, add or subtract **3000** to or from the *SIMODE* value.

SIPARMS item

The parameters for SUPERDEX access are defined in the screen, using standard **LET** statements.

Parameters

The dataset to access is defined in the *SIDSET* parameter, and the SI-path is defined in the *SIPATH* parameter. The *SIMODE* specifies how entries should be qualified and whether they should be returned in ascending or descending order. The search value, partial or generic key, range of values, or multiple search values are specified with the desired operators in the *SIARG* parameter.

The parameters of **SIPARMS** have the following meaning:

SIDSET The IMAGE name of the dataset. It must be enclosed in quotes and, if less than 16 characters, appended with a ; or single space.

SIPATH The name of the SI-path to access. It must be enclosed in quotes and, if less than 16 characters, appended with a ; or single space.

In performing successive **CALL DBGET...**s against multiple datasets, a common item used to logically link the files together (called the *SI-link*) may be additionally specified. It is required that the item assigned as the SI-link be configured as an SI-subkey in a concatenated SI-key; alternately, for SI-paths against a master dataset, it may be the IMAGE search field.

The SI-link is separated from the SI-path name by a comma, with the combined value terminated by a **SPACE** or **;** as shown:

SI-path,SI-link;

If performing a *projection*, which is used to logically link two files that do not contain a common item by reassigning the SI-link, the *SIPATH* parameter takes the form:

SI-path,new SI-link;

To locate entries that have been found by previous **CALL DBGET...**s in the active SI-subset, specify a value of **0;**.

Additionally, a length parameter (called the *SI-link-length*) may be assigned to the SI-link value as follows (please refer to *Section 4: Programming* of the **SUPERDEX User Manual** for further information):

SI-path,SI-link,SI-link-length;

- SIMODE** The *SIMODE* is the same as the *mode* parameter used in SUPERDEX's DBFIND intrinsic (see [Section 5](#) of the ***SUPERDEX User Manual***), but additional *modes* are also supported.
- SIMODE 0** Resets to sequential access. The *SIPATH* and *SIARG* parameters are ignored and do not need to be specified.
- SIMODE *n*+1000** By default, only entries that match the specified *SIARG* are returned. By adding 1000 to or subtracting 1000 from the *SIMODE*, all entries greater-than or equal-to the *SIARG* will be returned in ascending order.
- SIMODE *n*+2000** By default, entries are returned in ascending sorted sequential order. By adding 2000 to or subtracting 2000 from the *SIMODE*, entries will instead be returned in descending order.
- SIMODE *n*+3000** Same as *SIMODE* +1000, but returns entries less-than or equal-to the *SIARG* in descending order.
- SIARG** Value or values to search for. The *SIARG* value may contain all of standard capabilities, retrievals, and Boolean operators that are available in the standard SUPERDEX DBFIND.

SICOUNT item

In addition to the input parameters used in SIPARMS, the separate item **SICOUNT** is used as an output parameter and returns the number of entries which were qualified by **CALL DBGET ...** if called using *SIMODE* 1.

INDEX

A

ACCESS <> OK	2-8
Access methods	2-6
Approximate match retrieval	
QTP	2-8
QUICK	2-7
QUIZ	2-8
Ascending order	
Designer	3-21
QTP	2-6
QUICK	2-6, 2-20
QUIZ	2-6
AUTO=	2-28

C

Concatenated keys	2-6
Custom indexing	
QTP	2-10
QUICK	2-10
QUIZ	2-10

D

Definition	
SIARG	2-22, 2-23, 3-23
SIDSET	2-21, 3-22
SIMODE	2-21, 3-23
SIPATH	2-21, 3-22
Descending order	
Designer	3-21
QTP	2-6, 2-40
QUICK	2-6, 2-20
QUIZ	2-6, 2-40
Designer	
Ascending order	3-21
Descending order	3-21
Projection	3-22
Qualifying entries	3-21
SELECT	3-2
SI-Link	3-22
SICOUNT	3-24
SIMODE	3-21
SIPARMS	3-22

E

Errors	
SIQTP	2-45
SIQUIZ	2-45
Example	
QUICK	2-24

F

FILE	
SIPROC	2-3
FILE statement	
QUICK	2-18

G

Generic retrieval	
QTP	2-7
QUICK	2-7
QUIZ	2-7
GET statement	
QUICK	2-20
Greater-than retrieval	
QTP	2-8
QUICK	2-8
QUIZ	2-8
Grouped retrieval	
QTP	2-9
QUICK	2-8
QUIZ	2-9
grouping	
mismatched SI-key lengths	2-9
operation	2-9

I

INFO	
QUIZ	2-28
Invoking QUIZ	2-28

L

Less-than retrieval
 QTP..... 2-8
 QUICK..... 2-8
 QUIZ..... 2-8

M

Modifications
 Screens..... 2-18
 Multiple Criteria retrieval
 QUICK..... 2-9
 QTP..... 2-9
 QUIZ..... 2-9
 Multiple databases
 QUICK..... 2-19
 Multiple fields, sets, bases retrieval
 QTP..... 2-10
 QUICK..... 2-10
 QUIZ..... 2-10

O

OPEN_n 2-19

P

Partial retrieval
 QTP..... 2-7
 QUICK..... 2-7
 QUIZ..... 2-7
 POSTPATH..... 2-3, 2-6
 QUICK..... 2-18
 PowerHouse Overview..... 2-2
 Projection
 Designer..... 3-22
 PROMPT 2-36

Q

QTP 2-43
 Access methods 2-6
 Approximate match retrieval 2-8
 Ascending order..... 2-6
 Concatenated keys 2-6
 Custom indexing..... 2-10
 Descending order 2-6
 Generic retrieval 2-7
 Greater-than retrieval 2-8
 Grouped retrieval 2-9
 Less-than retrieval..... 2-8
 Partial retrieval 2-7
 Range retrieval..... 2-8
 Relational access
 Multiple criteria retrieval 2-9
 Multiple fields, sets, bases retrieval 2-10
 SIPATH 2-6
 Super-grouped retrieval 2-9
 QTP interface
 overview 2-5
 Qualifying entries
 Designer..... 3-21
 QUICK..... 2-20
 QUIZ..... 2-31
 QUICK
 ACCESS <> OK..... 2-8
 Access methods 2-6
 Approximate match retrieval 2-7
 Ascending order..... 2-6, 2-20
 Concatenated keys 2-6
 Custom indexing..... 2-10
 Descending order 2-6, 2-20
 Example screen 2-24
 FILE statement..... 2-18
 Generic retrieval 2-7
 GET statement 2-20
 Greater-than retrieval 2-8
 Grouped retrieval 2-8
 Less-than retrieval..... 2-8
 Multiple databases..... 2-19
 Partial retrieval 2-7
 POSTPATH 2-6, 2-18
 Qualifying entries 2-20
 Range retrieval..... 2-8
 Relational access
 Multiple Criteria 2-9
 Multiple fields, sets, bases retrieval 2-10
 Screen modifications 2-18
 SI-Link..... 2-21
 SICOUNT 2-23
 SIMODE..... 2-20

Q (continued)

QUICK

- SIPARMS2-18, 2-21
- SIPROC2-6
- Super-grouped retrieval.....2-9
- TRIM function2-20

QUICK interface2-17

- overview.....2-3

QUIZ.....2-40

- Access methods.....2-6
- Approximate match retrieval.....2-8
- Ascending order2-6
- CHOOSE2-40
- Concatenated keys2-6
- Custom indexing2-10
- DEFINE2-40
- Descending order2-6
- Examples.....2-41
- Generic retrieval.....2-7
- Greater-than retrieval2-8
- Grouped retrieval2-9
- INFO2-28

 - AUTO=2-28

- Less-than retrieval.....2-8
- Modifying source.....2-29
- Partial retrieval.....2-7
- Qualifying entries.....2-31
- Range retrieval2-8
- Relational access
 - Multiple criteria retrieval2-9
 - Multiple fields, sets, bases retrieval.....2-10
- SELECT.....2-40
- SIDEF2-4, 2-33

 - Expression.....2-34
 - Prompt text.....2-33
 - Search Criteria2-34
 - Variable.....2-33

- SIPATH.....2-4, 2-6, 2-35

 - Argument2-36
 - Dataset2-36
 - Multiple path retrieval.....2-37
 - Prompt text.....2-36
 - Search criteria2-36
 - SI-Link2-36
 - SI-Path.....2-36

- SORT2-40
- Super-grouped retrieval.....2-9

QUIZ interface

- overview.....2-4

- QTP.....2-8
- QUICK.....2-8
- QUIZ.....2-8

Relational access

- Multiple Criteria retrieval
 - QUICK.....2-9
 - QTP.....2-9
 - QUIZ.....2-9
- Multiple fields, sets, bases retrieval
 - QTP.....2-10
 - QUICK.....2-10
 - QUIZ.....2-10

Retrieval

- Sorted.....2-6

RXPROMPT2-36

S

Screen modifications

- QUICK.....2-18

SELECT

- Designer3-2

SI-Link.....2-6

- Designer3-22
- QUICK.....2-21

SIARG

- Definition.....2-22, 2-23, 3-23

SICOUNT2-3

- Designer3-24
- QUICK.....2-23

SIDEF2-4

- Expression2-34
- Prompt text2-33
- QUIZ.....2-33
- Search Criteria2-34
- Variable2-33
- Substrings2-34

SIDSET

- Definition.....2-21, 3-22

SIMODE

- Definition.....2-21, 3-23
- Designer3-21
- Mode 0.....2-21, 3-23
- Mode n+1000.....2-22, 3-23
- Mode n+2000.....2-22, 3-23
- Mode n+3000.....2-22, 3-23
- QUICK.....2-20

S (continued)

R

Range retrieval

- SIPARMS2-3
- Designer3-22

QUICK.....	2-18, 2-21
SIPATH.....	2-4, 2-6
Argument.....	2-36
Dataset.....	2-36
Definition.....	2-21, 3-22
Multiple path retrieval.....	2-37
Prompt text.....	2-36
QUIZ.....	2-35
Search criteria.....	2-36
SI-Link.....	2-36
SI-Path.....	2-36
SIPROC.....	2-3, 2-6
SIPROCN.....	2-19
SIQTP.....	2-5, 2-44
Errors and warnings.....	2-45
INFO=.....	2-5
Invoking.....	2-44
SIQUIZ.....	2-4, 2-38
Errors and warnings.....	2-45
EXIT.....	2-38
INFO=.....	2-4
Invoking.....	2-28
QUIT.....	2-39
USE.....	2-38
Sorted retrieval.....	2-6
Super-grouped retrieval	
QTP.....	2-9
QUICK.....	2-9
QUIZ.....	2-9

T

TRIM function	
QUICK.....	2-20

W

Warnings	
SIQTP.....	2-45
SIQUIZ.....	2-45

ENQUIRE

User Manual

Version 4.2

All updates to or derivatives of the SUPERDEX™ and ENQUIRE™ computer software provided herein are copyrighted and may not be copied except for archival purposes, to replace a defective copy, or for program error verification by Licensee. Copyrighted material may not be copied onto any media (e.g. magnetic tape, paper tape, disc memory cartridges, read-only memory, etc.) for any other purposes. The authorization to duplicate copyrighted materials hereunder shall not be construed to grant the Licensee or Licensee's customer the right to use copyrighted SUPERDEX or ENQUIRE material in any manner other than which is provided in this agreement or otherwise approved in writing by Dr. Wolfgang Matt or Bradmark Technologies, Inc.

© 1988, 2000 Bradmark Technologies, Inc.

All rights reserved

Printed in the U.S.A. (October 2000)

Adager is a trademark of Adager

AdvanceLink, Business Basic, HP, IMAGE, TRANSACT, TurboIMAGE, and TurboIMAGE/XL are trademarks of Hewlett-Packard Company

ASKPLUS and VISIMAGE are trademarks of ARES

Business Session is a trademark of Tynmlabs Corporation

dBASE, dBASE III, and dBASE III Plus are trademarks of Ashton-Tate Corporation

DBGENERAL is a trademark of Bradmark Technologies, Inc.

DIF is a registered trademark of Software Arts Products

ENQUIRE and SUPERDEX are trademarked product names of Bradmark Technologies, Inc. for the SI-IMAGE and ENQUIRE packages developed and implemented by Dr. Wolfgang Matt

Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation

Macintosh is a registered trademark of Apple Computer, Inc.

MPEX/3000 is a trademark of Vesoft Inc.

PowerHouse, QUIZ, QUICK, and QTP are registered trademarks of Cognos Incorporated

Reflection is a registered trademark of Walker Richer & Quinn, Inc.

SPEEDWARE is a trademark of Speedware Corporation

SYDAID is a trademark of Sydes

TaskMaster is a trademark of Fransen/King

About this manual

In writing this manual, we assume that you have working knowledge, although not internal knowledge, of IMAGE and the HP3000.

All references to IMAGE in this manual and throughout the SUPERDEX and ENQUIRE packages also apply to TurboIMAGE and TurboIMAGE/XL unless otherwise noted.

This manual is arranged in the following format:

Section 1 provides an *Overview* of the ENQUIRE package, its features, capabilities, and benefits.

Section 2 describes ENQUIRE *Operation*, including installation, functions, redirecting output, invoking the program, and function key assignments.

Section 3 discusses utilizing data dictionaries for each database, referred to as *Database profiles*.

Section 4 reviews the procedures for defining, modifying, deleting, copying, and executing *Search profiles*, the heart of ENQUIRE functionality. Included are various methods of reporting data to the screen and printer.

Section 5 describes how to use ENQUIRE from other HP3000 and PC programs.

Section 6 shows various facilities for *Customizing ENQUIRE*, including custom-defined VPLUS forms and message catalogs, which facilitate Native Language Support.

Appendix A discusses considerations for *Database structural changes*, their impact on ENQUIRE database and search profiles, and how to restore consistency when the structure of a database accessed by ENQUIRE is modified.

Appendix B examines ENQUIRE's *Internal structures*, including the structure of the DBENQ database that holds database and search profile configuration and determining the access method used by ENQUIRE when executing a search profile.

Appendix C documents ENQUIRE's *Maximum limits*.

Appendix D lists all of ENQUIRE's *Error and exceptional conditions* with messages, their meanings, and recommended actions.

Table of contents

Section 1: Overview.....	1
Why ENQUIRE?	2
How ENQUIRE works.....	4
The database profile	5
The search profile	5
Security	6
Access methods.....	6
Screen formats.....	7
Native language support.....	7
Online help facility	7
Batch facility	7
Access methods.....	8
Multiple keys in master and detail datasets	8
Concatenated keys containing multiple fields	8
Sorted sequential retrieval.....	8
Keyword retrieval	9
Generic and partial key retrieval.....	9
Greater-than, less-than, and range retrieval	9
Grouped retrieval	9
Super-grouped retrieval.....	10
Relational access: multiple criteria retrieval.....	10
Relational access: multiple fields, sets, bases.....	10
Custom SI-path access	10
Section 2: Operation.....	2-1
Installation	2-1
ENQUIRE functions	2-1
Redirection to other groups/accounts	2-1
Redirecting output to printer.....	2-2
Invoking and using the ENQUIRE program.....	2-2
Function key operation.....	2-3
Section 3: Database profiles	3-1
Defining a new database profile	3-2
Main menu	3-2
Defining the database and its password.....	3-3
Defining the database profile password.....	3-3
Selecting the database or a specific dataset to configured.....	3-4
Editing and saving the current form.....	3-4
Selecting the dataset to configure	3-4
Defining input and output field attributes.....	3-5
Defining prompts	3-6
Defining headings	3-6

Defining edit mask specifications	3-6
Defining decimal point position	3-8
Completing the database profile	3-8
Modifying or deleting a database profile	3-9
Modifying a database profile	3-9
Deleting a database profile	3-10
Section 4: Search profiles	4-1
Defining a new search profile	4-2
Main menu	4-2
Defining databases and their passwords	4-4
Forms specification	4-4
Output format	4-4
Defining the search profile password	4-5
Editing and saving the current form	4-5
Defining datasets to access	4-6
Defining fields for selection	4-7
Defining fields for selection	4-8
Defining SUPERDEX SI-path access	4-10
Defining fields for output	4-11
Defining input and output field attributes	4-13
Completing the search profile	4-14
Modifying or deleting a search profile	4-15
Modifying a search profile	4-15
Deleting a search profile	4-17
Copying an existing search profile	4-18
Executing a search profile	4-20
Main menu	4-20
Specifying values to search for	4-21
Searching for all values	4-22
Searching for partial values	4-23
Searching for generic values	4-23
Searching for greater-than/less-than values	4-24
Searching for not-equal-to values	4-24
Searching for a range of values	4-24
Searching for multiple values in a single field	4-25
Searching for values that contain reserved characters	4-25
Searching for negative numeric values	4-26
Searching compound IMAGE items	4-26
Specifying limit on number of entries to return	4-26
Specifying file for output	4-26
Reporting	4-28
Reporting number of qualifying entries	4-28
Reporting entries to screen (LIST ALL)	4-28
Reporting totals only to screen (LIST SUMS)	4-30
Printing entries on printer (PRINT)	4-31
Storing entries to file (STORE)	4-32

Section 5: Batch and micro interfaces	5-1
Batch interface	5-2
Creating an input file	5-2
Executing a search profile in batch mode	5-3
Executing a search profile from another program	5-3
Micro interface	5-4
Automatically downloading store files to microcomputer	5-4
Manually downloading store files to microcomputer	5-4
Executing a search profile from a Reflection command file	5-5
Lotus interface.....	5-6
Controlled Interface.....	5-8
Section 6: Customizing ENQUIRE.....	6-1
Custom forms	6-2
Custom forms	6-2
Custom menus	6-2
Custom input forms	6-2
Custom input forms	6-3
Custom output forms.....	6-3
Custom help forms	6-4
Native language support.....	6-5
Setting the native language	6-5
Forms file (with Native Language Support)	6-6
Error and status message catalog (with Native Language Support)	6-6
HPFD support	6-7
Appendix A: Database structural changes.....	A-1
Appendix B: Internal structures.....	B-1
The DBENQ database.....	B-1
Determining ENQUIRE access method	B-1
Appendix C: Maximum limits.....	C-1
Appendix D: Error and exceptional conditions.....	D-1
ENQUIRE error, exceptional, and status messages.....	D-1

SECTION 1:

Overview

Overview

The ENQUIRE package provides a convenient means for locating entries in any database using several advanced methods and outputting them in various formats, including screen display, printed report, Lotus 1-2-3, dBASE, and other microcomputer formats.

Chapter 1 Why ENQUIRE?

Function explains the basic benefits and capabilities of ENQUIRE.

Chapter 2 How ENQUIRE works

Function defines concepts and overall features.

Chapter 3 Access methods

Function used to qualify entries using ENQUIRE, both with and without SUPERDEX.

CHAPTER 1:

Why ENQUIRE ?

ENQUIRE was created to perform powerful retrievals against IMAGE, TurboIMAGE, and TurboIMAGE/XL databases in a very quick, user-friendly manner using a different philosophy and methodology than QUERY/3000 and other reporting tools.

Unlike other packages, ENQUIRE separates the specification of the inquiry parameters (the databases and datasets to access, fields to search on, fields to output, etc.) and of the search criteria (actual data values) into two distinct operations. In ENQUIRE, a *search profile* that describes an inquiry is created in *definition* mode, and is then invoked with search criteria specified in *execution* mode.

For example, a search profile could be defined that finds customers in a CUSTOMER-MASTER dataset with balances over a certain amount which have been outstanding for more than a certain number of days, and then outputs the associated invoice numbers, dates, and amounts from an INVOICE-HEADERS dataset. Upon executing this search profile, the user would be prompted for and specify the amount and number of days outstanding, and the qualifying entries would be retrieved.

Because end users supply only the search values in response to pre-defined prompts, they do not require any knowledge of database structure or need to remember any commands or syntax; they just access the search profile by name and fill in the blanks.

Search profiles are typically defined by data processing personnel by specifying the database(s) to access (up to 4) and then "checking off" the datasets to access (up to 16), fields to select on (up to 16), and fields to report (up to 128). This process is simple enough to be done by even a regular user.

In defining search profiles, only the "what" (bases, sets, fields) is specified--the "how" is figured out by ENQUIRE automatically. ENQUIRE chooses the quickest, most efficient database access methods, whether using IMAGE paths, SUPERDEX paths, or serial reads. SUPERDEX paths provide the fastest and most flexible access, although ENQUIRE does permit limited SUPERDEX-type access (partial key, generic key, greater-than/less-than, and range retrievals) on IMAGE keys!

To make ENQUIRE search profiles more user-friendly and output more meaningful, custom prompts, header titles, and edit masks may optionally be defined for any field. To speed up this process, a *database profile* may be created for each database to define these attributes globally, thereby acting as a data dictionary.

Default input and output VPLUS forms are provided, and may be replaced with custom forms where desired. Additional customizing may be done by redefining error and status messages. These facilities are also used in ENQUIRE's Native Language Support.

Once users locate the desired entries in execution mode, they may display them on the screen, print them, or output them to a file in a pre-defined format for further use on the HP3000 or for transfer to a microcomputer. If using Reflection, ENQUIRE can automatically download files from the HP3000 to the microcomputer simply by specifying a microcomputer file name for output. ENQUIRE can output data in various popular microcomputer formats for use by dBASE, Lotus, various word processors, and other programs on PCs and Macintosh.

In addition, if you see that ENQUIRE can solve a lot of your reporting needs but wish you could provide its capabilities from within your own programs, no problem. ENQUIRE's *batch facility* allows search profiles to be executed with specified selection values by handling ENQUIRE as a son process and passing it the required parameters. This same facility can also be utilized from HP3000-to-PC transfer program command files (such as those supported by Reflection) to transfer files output in microcomputer format to micros.

We know that you'll find ENQUIRE to be truly user-friendly, both in defining and executing search profiles. And coupled with its support of SUPERDEX access methods and popular output formats, you'll be amazed at its speed, power, simplicity, and usefulness.

CHAPTER 2:

How ENQUIRE works

The **ENQUIRE** program provides a powerful, user-friendly facility in which various data retrieval screens called *search profiles* are defined to perform specific inquiries against:

- up to 16 fields, in
- up to 16 datasets, in
- up to four databases

The search profile configuration is entirely screen-driven and requires that the bases, sets, and fields to access be defined. It also requires that for key fields the desired access method (IMAGE, SUPERDEX, or sequential) be specified; if multiple SI-paths exist for a selection field, the preferred key must be indicated.

The search profiles do not contain the data values to search for; rather, the search criteria are specified at execution time and may include:

- partial keys
- generic keys
- greater-than/less-than constructs
- not-equal-to constructs
- ranges
- values for multiple fields
- multiple values for a field (boolean operators)

For searches against fields that are configured as SUPERDEX SI-keys, the following may additionally be selected on:

- for keyworded SI-paths, any significant word contained in the SI-key
- for grouped SI-paths, any value in any SI-key in the group
- multiple values for a field, using boolean operators

Up to 128 fields related to each entry may be output in various formats:

- displayed on the screen
- printed
- written to a file in binary format
- written to a file in Lotus 1-2-3 format for use on a microcomputer
- written to a file in dBASE format for use on a microcomputer
- written to a file in comma/quote-delimited format for use on a microcomputer
- written to a file in tab-delimited format for use on a Macintosh

While the ENQUIRE package is designed for use with SUPERDEX, it does not require that a database be configured for SUPERDEX to operate. ENQUIRE will use IMAGE paths, SI-paths,

sequential searches, or a combination of these to access data. The desired access method for each selection field is specified in the search profile configuration.

Additionally, the capability of defining data dictionaries for all databases exists, permitting global input and output field attributes to be defined once for each database and overridden for specific search profile requirements. The dictionary for each database is referred to as a *database profile*.

The database profile

A database profile may optionally be defined for an entire database or for selected datasets within a database, permitting global attributes to be configured once and then automatically utilized for all search profiles that access that database when they are executed.

Default database-profile parameters may be overridden for any search profile when defined. It is recommended that database profiles be created in their entirety before defining any search profiles to set the desired default parameters.

Database profiles are created in ENQUIRE's *definition mode* (accessible by using the **DEF** entry point) and saved in a special internal database (named **DBENQ**). They may also be modified or deleted in this mode.

Each database profile is assigned the name of the database it profiles and includes several optional attributes:

- the prompts used to identify each item on input
- the headings used to identify each item on output
- the decimal point position for numeric fields
- edit masks for alpha and numeric fields

The search profile

The search profile is also created in ENQUIRE's definition mode and saved in the DBENQ database, and may also be modified or deleted in this mode.

The search profile does not contain the values to search for--these are entered when the search profile is executed by an authorized user in *execution mode*.

In the definition mode, each search profile is assigned an arbitrary name, and its *input format* and *output format* specifications are defined.

The input format specification includes:

- the search profile password
- the database(s) to access
- the dataset(s) to access
- the field(s) to search
- the type of access (SUPERDEX, IMAGE, or sequential)
- if multiple SI-paths exist for a selection field, which one to use
- the prompts used to identify each item

The output format specification includes:

- the field(s) to display
- the heading(s) used to identify each item
- any edit masks for alpha and numeric fields
- the decimal point position for numeric fields
- the output format (screen, file, special format)

Input and output field attributes default to those defined in the database profile and may be overridden. If not defined in a database profile, they default to their item names with no edit masks or decimal values.

Security

In defining and modifying database and search profiles, access to sets and items is restricted by the IMAGE user class that corresponds to the database password specified. Sets and items to which access is restricted may not be included in a search profile, nor are they displayed for selection.

Each database and search profile may optionally be protected by an arbitrary eight-character password that restricts execution to only those users who supply the correct password. This password may be changed as desired by modifying the database or search profile.

Access methods

In defining a search profile, the bases, sets, and items to access are specified. Additionally, the method of access (IMAGE, SUPERDEX, or sequential) must be specified for selection fields. For selection fields that are indexed by more than one SI-path, the SI-path to use must be indicated.

In executing a search profile, only the data values are specified. ENQUIRE will use the configured access method for each selection field for which a value is specified. For IMAGE keys configured for IMAGE access in which a value other than a full key value is specified, a sequential read is performed, because IMAGE does not support partial-key retrieval.

Screen formats

All the VPLUS forms used for defining and executing search profiles are generated by default by ENQUIRE, but may alternatively be custom-defined in FORMSPEC.

ENQUIRE operates entirely in block mode, and therefore must be run from a terminal or microcomputer that supports block mode.

Native language support

All forms and message catalogs are supplied in several different languages, which may be selected by a :RUN... ;PARM or via file equations, as explained in the *Operation* section.

Because all screens used in ENQUIRE are VPLUS forms, these forms may easily be adapted to any native language. Message catalogs are in GENCAT format, and may also be easily modified in any editor.

Refer to the *Customizing ENQUIRE* section for instructions for setting the native language for your installation, as well as modifying the message catalog and VPLUS forms.

Online help facility

For each input and output form in ENQUIRE there is a corresponding help form which provides explanatory information. The corresponding help form may be displayed from any regular form by selecting the **F1** function key, which is labelled **HELP**. To return to the regular form, select the **F8** function key.

Batch facility

ENQUIRE search profiles are normally executed online but may alternatively be run from batch jobs with pre-defined search values using ENQUIRE's batch facility.

This same technique may be used to enable ENQUIRE search profiles to be executed from within application programs, and for invoking ENQUIRE from within HP3000-to-PC transfer package command files, such as those supported by Reflection.

The batch facility is invoked by running ENQUIRE with its \$STDIN redirected to a specially-formatted file containing the search profile name, search criteria, and other parameters.

CHAPTER 3:

Access methods

This section overviews the various methods available in ENQUIRE (with and without SUPERDEX) to access data in IMAGE databases.

Multiple keys in master and detail datasets

Up to 16 fields in any dataset may be defined for selection in a search profile, which may be in a single dataset or multiple datasets.

For a selection field that is an SI-key and specified for SUPERDEX access, ENQUIRE performs an indexed-sequential retrieval via SUPERDEX. For IMAGE keys specified for IMAGE access, ENQUIRE performs either a keyed or chained read. A serial read is performed for any field that is not a key in either IMAGE or SUPERDEX, or which has been explicitly marked for sequential access.

Concatenated keys containing multiple fields

One selection field is defined for each SI-subkey in the SI-key and the values are recombined by ENQUIRE. Not all SI-subkeys need to be included as selection fields--ENQUIRE will perform a retrieval using the SI-path regardless.

Concatenated SI-keys are transparent in ENQUIRE retrievals. Whichever SI-subkeys are included as selection fields in the search profile are prompted for individually and internally combined by ENQUIRE.


Access is quickest when all SI-subkeys in the SI-key are used as selection fields. If this is not the case, it is desirable to have at least the first SI-subkey used as selection field; the more SI-subkeys thereafter are included, the more efficient the access.

Sorted sequential retrieval

Entries are unconditionally returned in sorted sequential order for entries qualified in indexed access mode along an SI-path. To read all the entries in a set in ascending order, specify a blank selection value.

If IMAGE access is performed, entries are returned in either chained or sequential order, based on the type of IMAGE access performed.

Keyword retrieval

 **Keyword retrieval is available only in the SUPERDEX II package.**

Any selection field that is configured as a keyworded SI-key or the first SI-subkey in a concatenated SI-key is treated as keyworded.

A comparison is made based only on the keyword length configured for the SI-path.

Generic and partial key retrieval

Partial key retrieval may be performed against IMAGE fields or SI-keys by specifying the partial key value appended with an @ (e.g. **HEWL@**). This will locate all entries that match on the significant characters followed by anything.


Greater-than, less-than, and range retrieval

These types of retrievals may be performed against any IMAGE field or SI-key and are facilitated by embedding special operators in the search value for any field.

Greater-than retrieval is accomplished by prefixing the search value with the **>** operator (e.g. **>1000**): greater-than-or-equal-to retrieval is done using the **>=** operator. Less-than retrieval uses the **<** operator, and less-than-or-equal-to retrieval uses the **<=** operator as a prefix, and not-equal-to retrieval uses the **<>** operator.

Range retrievals are performed by using the **:** operator between two selection values. For example, a range search to find all the entries with amounts between 500 and 1000, inclusively, is specified with **500:1000**.

Grouped retrieval

 **Grouped retrieval is available only in the SUPERDEX II package.**

Grouped retrieval may be performed against grouped SI-paths. If the field that is the first configured SI-key in the group is used as a selection field, all SI-keys in the group are automatically searched when the search profile is executed.

A comparison is made based only on the SI-key length configured for the SI-path.

Super-grouped retrieval



Super-grouped retrieval is available only in the SUPERDEX II package.

For retrieval against super-grouped SI-paths, ENQUIRE will use the super-group to perform the retrieval but will restrict the output to only those datasets that were selected for output in the search profile.

In configuring the search profile for super-group access, select the master set to be accessed before its related detail sets.

Relational access: multiple criteria retrieval



Relational access is available only in the SUPERDEX II package.

ENQUIRE supports retrieval by multiple criteria by allowing multiple values to be specified for a single field. The Boolean operations AND, OR, and AND NOT may be performed against the multiple values by delimiting the values with the operators +, /, and - respectively.

Relational access: multiple fields, sets, bases

Relational queries may be performed across multiple fields, datasets, and databases in ENQUIRE. ENQUIRE permits up to 16 fields in up to 16 datasets in up to four databases to be involved in a search profile.

The bases to access are defined on one form, and then additional forms are completed for each database/dataset combination. This is done for both input and output formats. There must be a physical or logical linkage between datasets and databases. ENQUIRE will determine, when the search profile is defined, if a linkage can be made and will reject the attempted configuration and display an error message if it cannot. ENQUIRE internally performs all lookups necessary to complete the retrieval.

Entries must match on all specified search values and exist in all datasets and databases in order to qualify.

Custom SI-path access

Retrievals may be performed using custom SI-paths (created via the SIUSER procedure). If any custom SI-paths have been defined for a selected dataset, ENQUIRE displays the literal "**CUSTOM INDEX**" for selection in order to perform retrieval by the SIUSER path.

SECTION 2:

Operation

Installation

This section assumes that you have already loaded SUPERDEX on your system, as described in the separate *SUPERDEX loading instructions*.

ENQUIRE functions

The ENQUIRE program performs several functions against database profiles and search profiles:

- define a new database profile or search profile
- modify an existing database profile or search profile
- delete an existing database profile or search profile
- execute an existing search profile

These functions are described in the following chapters.

Redirection to other groups/accounts

ENQUIRE's internal control files (as well as the ENQUIRE program) are by default contained in SUPERDEX.SYS. The files are:

- **DBENQ** database in which database and search profiles are saved
- **FOENQ nnn** VPLUS forms file containing configuration, input, and output forms
- **ERENQ nnn** error and status message catalog

where nnn is the language ID number (*langid*) of the native language used (Native-3000 is 000). Refer to the *Customizing ENQUIRE* section for information about running ENQUIRE in other languages).

If these files are moved to or duplicated in another group or account, it is necessary to issue fully-qualified :FILE equations to redirect access, as shown:

```
:FILE DBENQ.PUB.SUPERDEX=DBENQ.mygroup.myacct
:FILE FOENQ000.PUB.SUPERDEX=FOENQ000.mygroup.myacct
:FILE ERENQ000.PUB.SUPERDEX=ERENQ000.mygroup.myacct
```


In ENQUIRE, databases are specified by their unqualified names (without group and account), so a file equation is required for databases that do not reside in the logon group and account. For example:

```
:FILE OEDB=OEDB.DATA.SALES
:FILE CUSTDB=CUSTDB.DATA.SALES
```

would be issued if the OEDB and CUSTDB databases reside in DATA.SALES. This provides a convenient means for alternating between test and production databases when developing database and search profiles.

Redirecting output to printer

Entries printed via ENQUIRE's print function are by default printed on the system line printer (device class **LP**). You may select an alternate printer by issuing a file equation to it for the formal file designator **ENQLIST**. For example:

```
:FILE ENQLIST;DEV=LASER
```

would direct all printed output to the printer that corresponds to device class **LASER**.

A printer other than device class LP may alternately be selected by specifying it in the Output file name field of the output form (refer to the *Reporting* chapter of the *Search profiles* section for details).

Invoking and using the ENQUIRE program

By default, the ENQUIRE program is contained in SUPERDEX.SYS and may be run in one of three ways. To define, modify, or delete, or execute database and search profiles, run ENQUIRE with the **DEF** entry point:

```
:RUN ENQUIRE.PUB.SUPERDEX,DEF
```

This method is for use in developing search profiles and is not intended for end users.

To execute existing search profiles:

```
:RUN ENQUIRE.PUB.SUPERDEX
```

This is intended for end users, since it does not permit access to development features. This method allows users to simply enter the name of the search profile they wish to execute.

The third method of executing ENQUIRE is with the MENU entry point:

```
:RUN ENQUIRE.PUB.SUPERDEX,MENU
```

This entry point allows the user to see those profiles that they can execute, and can simply mark the search profile they want with an X. This method does not require the user to remember the names of the various search profiles available to him/her, but does require some minor setup.

Included in the FOENQ000 forms file are two special forms; MENU and MENU_____H (the MENU and HELP screen (the help screen name must be 15 characters in length), respectively). These forms can be modified (using FORMSPEC) to include the desired search profiles for a given user. The only requirement is that the field identifying each search profile on the FORM must contain the name of the search profile to be executed.

Remember, ENQUIRE operates in block mode: press **TAB** (not RETURN) to go forward one field and **SHIFT + TAB** (simultaneously) to go back one field. You may alternatively use the cursor keys. Press **ENTER** (NOT carriage return) when you are finished with a form.

Function key operation

Throughout ENQUIRE, function keys are consistently available to provide various actions. When in definition mode, the function keys are defined as follows:

Definition Mode Function Keys		
Key	Label	Description
F1	HELP	Displays corresponding (context-sensitive) help form
F2	PREVIOUS PAGE	Go back to previous page of multi-page form
F3	NEXT PAGE	Skip ahead to next page of multi-page form
F4	PRINT	Print the current form
F5	PREVIOUS FORM	Go back to previous definition form
F6	NEXT FORM	Skip ahead to next definition form
F7	SAVE CURRENT	Save profile as defined thus far
F8	MENU	Go back to Main Menu, flush changes

In execution mode, the following function keys are available on the input selection form:

Execution Mode: Input Selection Function Keys		
Key	Label	Description
F1	HELP	Displays corresponding (context-sensitive) help form
F2	LIST ALL	Display all qualifying entries
F3	LIST SUM	Total and display sum of similar entries
F4	PRINT	Output results to specified or default printer
F5	STORE	Output results to specified file in configured format
F6		(Not Defined)
F7		(Not Defined)
F8	MENU	Go back to Main Menu

In execution mode, the following function keys are available on the output selection form:

Execution Mode: Output Selection Function Keys		
Key	Label	Description
F1	HELP	Displays corresponding (context-sensitive) help form
F2	PREVIOUS ENTRIES	Re-display previous screen of qualifying entries
F3	NEXT ENTRIES	Display next screen of qualifying entries
F4		(Not Defined)
F5	PREVIOUS PAGE	Re-display previous screen of fields for the current entry
F6	NEXT PAGE	Display next screen of fields for the current entry
F7	SELECT/ OUTPUT	Enter a new search selection, or output qualifying entries
F8	MENU	Go back to Main Menu

SECTION 3:**Database profiles**

This section describes *database profiles*, which serve as data dictionaries that are accessed when defining search profiles.

Database profiles serve as data dictionaries for search profiles that are subsequently defined and executed. They are optional, and each contains global input and output field attributes for a database or for selected datasets within a database.

Attributes configured in a database profile are automatically applied to all search profiles as they are created, eliminating the need to re-specify these attributes when defining each search profile. It is possible, however, to override the database profile attributes in any search profile.

Database profiles are accessed when defining search profiles and define the default parameters for each search profile that accesses the database. Database-profile parameters may be overridden by any search profile. It is recommended that database profiles be created in their entirety before defining any search profiles to set the desired defaults.

Chapter 1 Defining a new database profile

Function reviews the process of *Defining a new database profile* and its attributes, including item headings, labels, and edit masks.

Chapter 2 Modifying or deleting a database profile

Function discusses the methods of *Modifying or deleting a database profile*.

CHAPTER 1:

Defining a new database profile

Main menu

To define a new database profile, run ENQUIRE with the **DEF** entry point, and then enter **B** in the Option box. Then enter the unqualified name of the database (without group and account--a file equation must already be set for databases that do not reside in the logon group an account) to profile, and leave the password blank (if you enter the name of an existing database profile, ENQUIRE will automatically go into modify/delete mode rather than add mode).

```
SUPERDEX/ENQUIRE : Main Definition Menu

Select an option ..... B   D = define new Search profile or
                             modify existing Search profile

                             B = define new Database profile or
                             modify existing Database profile

                             C = copy existing Search profile under
                             new name and modify

                             X = execute existing Search profile

Profile name ..... OEDB_____
password ..... [           ] (existing profile only)

New profile name ..... _____ (Copy option only)
```

Once both fields have been specified, press **ENTER** to process the screen's contents.

After completing the Main Menu, a second form is issued which defines:

- database name
- database password
- database profile password
- whether to delete or modify the current database profile
- whether to define all items in the base or only in specific sets

```

SUPERDEX/ENQUIRE : Define Database Profile Attributes

Database name ..... OEDEB
password ..... [      ]

Profile password ... _____ (optional)

Delete ? ..... _      Y = delete this database profile
                       N = modify this database profile

Scope ..... _      G = all items in database
                       S = items in specific dataset(s)

```

Defining the database and its password

The database name is predisplayed. Enter a database password below the database name. Ensure that the password specified grants read access to all the datasets and items that are required for the database profile, as only those sets and items will be displayed and available for selection; inaccessible sets and items are restricted and therefore not displayed. If you are logged on as the database creator, you may enter the creator password (;).

The password is not echoed to the screen for security reasons.

Defining the database profile password

Access to the database profile may optionally be restricted by assigning an arbitrary password of up to eight characters to the profile. This password must be specified when attempting to modify or delete the profile.

Selecting the database or a specific dataset to configured

The recommended method is to first define attributes for all items in the database, and then to override them as necessary for specific fields in specific datasets. For example, if the item NAME is used in both the CUSTOMERS and VENDORS datasets, different labels and headings (e.g. Customer name and Vendor name) should be defined for each to avoid confusion for users when executing associated search profiles.

G is predisplayed which may be overwritten by **S**.

Editing and saving the current form

You may edit the form further by skipping forward from field to field with the TAB key or backward with the SHIFT and TAB keys in combination.

Once the form has been completed, press the **ENTER** key to save it in ENQUIRE's internal database, or a function key to perform a different action.

Selecting the dataset to configure

If **S** was typed in (i.e. that items in a specific dataset should be profiled), a separate dataset definition form is displayed. This form is used to select which dataset to define attributes for.

```
SUPERDEX/ENQUIRE : Define Dataset for Formats and Labels      base: OEDB
-> Select one dataset by marking it with an X :
_ M CUSTOMERS                _ M ORDER-HEADERS                _ D ORDER-LINES
```

Each dataset is indicated by name and prefixed with a code defining the dataset type (**A**=automatic master, **M**=manual master, **D**=detail). Datasets are listed in the order in which they appear in the base's dataset list.

Up to 48 datasets are displayed on one screen; additional sets are shown on one or more additional pages that are automatically displayed when **ENTER** is pressed after completing the current page. You may skip between the pages by using the **F2** and **F3** function keys.

Select the dataset to be configured by entering an **X** in the adjacent box, and press **ENTER** to proceed to the next form or a function key to perform the desired action. As you can only select one dataset at a time, repeat the action if you wish to configure more datasets.

Defining input and output field attributes

An input/output field attribute definition form is displayed for either all items in the database or in the specified dataset. The base name or base and set names are indicated in the upper right corner. This form defines:

- the prompt to display for each input field
- the heading to display for each output field
- edit mask specifications for alphanumeric and numeric fields
- the decimal point position for numeric fields

SUPERDEX/ENQUIRE : Define Labels & Formats base/set : OEDB /				
Item	Prompt	Heading	Edit spec	Decimal
N	CUSTOMER-NO			-
C	CUSTOMER-NAME			-
C	ADDRESS-1			-
C	ADDRESS-2			-
C	CITY			-
C	STATE			-
N	ZIP-CODE			-
N	PHONE-AREA-C			-
N	PHONE-PREFIX			-
N	PHONE-SUFFIX			-
C	CUSTOMER-ABBR			-
N	ORDER-NUMBER			-
C	ORDER-TYPE			-
N	ENTRY-DATE			-
C	PO-NUMBER			-
N	BRANCH-LOCAT			-

For an entire database, each item is indicated by name and listed in the order of the database item list. For a dataset, each field is indicated by its item name and listed in the order of the dataset item list.

Defining prompts

By default, in executing a search profile each selection field value is prompted for by its item name. Alternatively, a different *prompt* of up to 20 characters may be defined to label each field. This feature is especially useful for item names that are not self-explanatory.

Defining headings

A *heading* is displayed in printed reports and included in some store formats. By default, in executing a search profile, each output field is headed by its item name. Alternatively, a different heading of up to 20 characters (or up to 10 characters if the search profile is configured to store in DBF format) may be defined for each field. If a heading exceeds the field length, it is broken into two lines when reported.

This feature is also especially important for item names that are not self-explanatory.

Defining edit mask specifications

Various *edit masks* may be defined for reformatting alphanumeric and numeric values when they are displayed, such as adding dollar signs or decimal points to dollar amounts.

These edit masks are the same as those used in QUERY/3000's REPORT facility, with the addition of a *list sums* option. The following characters can be used in various combinations in one or more positions in the edit mask for each item or field:

X Any character (alphanumeric items only)

For alphanumeric items (data types X or U), each **X** represents one character in each corresponding position in the output field. Enough **X**s to cover the length of the field should be specified; otherwise, the value will be truncated. For example, the edit mask **X-X-X-X** would reformat the value "ABCD" as A-B-C-D but the mask **XX** would reformat it as AB.

9 Any numeric character (numeric items only)

For numeric items (data types I, J, K, P, R or Z), each **9** represents one number in each corresponding position in the output field.

Z Blank zero-replacement character (numeric items only)

Same as **9**, but leading zeroes are replaced with blanks.

***** Asterisk zero-replacement character (numeric items only)

Same as **Z**, but leading zeroes are replaced with ***** s.

\$ Dollar sign (numeric items only)

Same as **Z**, but the first zero in the field is replaced by a **\$**.

- **Decimal point (numeric items only)**
Position of the decimal point; can only appear once in an edit mask.
- CR Credit sign (numeric items only)**
The characters **CR** are displayed in the two rightmost positions in the mask for negative values, immediately following the value. For positive values, two blanks are displayed instead.
- **Negative sign (numeric items only)**
Same as **CR**, but instead a **-** is displayed in the rightmost position for negative values; for positive values, one blank is displayed instead.
- x Literal insertion character (alphanumeric or numeric items)**
Any printable ASCII character (except the reserved characters **x**, **9**, **z**, *****, **\$**, **CR**, and **-**) is displayed literally in its corresponding position in the edit mask. Any insertion character appearing in the edit mask to the left of the leftmost significant digit of the value is replaced with a blank or asterisk (depending on the zero-replacement character specified).
- S List sums (numeric items only)**
Causes ENQUIRE to total the value of this field for all similar entries and display the total on a single line rather than displaying each entry if the LIST SUMS display format is selected. **S** must be specified in the first position of the edit mask.

The following tables show examples of edit masks and their effects on Character (data types U and X) and Numeric (data types I, J, K, R, P, and Z) item values:

<i>Alphanumeric value</i>	<i>Edit mask</i>	<i>Result</i>
ABCD	x-x-x-x	A-B-C-D
A34B		01/31/88 - - - A

<i>Numeric value</i>	<i>Edit mask</i>	<i>Result</i>
0059	\$\$\$,999	\$059
001024	zzz ,zzz	1,024
-0010555	\$\$,\$\$\$.99CR	\$105.55CR
00010555	\$\$,\$\$\$.99CR	\$105.55
-0010555	\$\$,\$\$\$.99-	\$105.55-
15039250	\$,\$\$\$,\$\$\$.99CR	\$150,392.50
00049	*****	***49
044240474	999-99-9999	044-24-0474
-2145	\$,\$\$\$\$.99	\$21.45
209 and 10 on two entries	s	219+

Defining decimal point position

Because numeric item values are stored with no decimal point, the decimal point position should be specified for numeric items that contain decimal values (excluding items of data type R). This is done in the *decimal* field.

The decimal point position is used in both input and output formats. Microcomputer programs generally assume that the decimal point is fixed, so it must be specified here. If no decimal point is specified (either in this field or in the edit mask), 0 is assumed and the decimal point is suppressed.

If an edit mask that includes a decimal point is specified, the decimal field value must correspond to the position of the decimal point in the edit mask.

Completing the database profile

Up to 16 fields are shown on a single page. If the base or set contains more than 16 items or fields, the other fields are displayed on one or more additional pages which are automatically displayed when **ENTER** is pressed after completing the current page. You may skip between the pages with the **F2** and **F3** function keys.

Once all the items or fields in a base or set have been configured, press **ENTER** to save the attributes and return to the previous form or to the menu (or press a function key to perform a different action).

CHAPTER 2:**Modifying or deleting a database profile****Modifying a database profile**

Existing database profiles may be modified in very much the same way and using the same forms as when defining a new database profile. Refer to the *Defining a new database profile* chapter for a discussion.

To modify an existing database profile, run ENQUIRE with the **DEF** entry point, then specify **B** in the Main Menu Option box and enter the name and password of the database profile (if one was assigned), as shown:

```
SUPERDEX/ENQUIRE : Main Definition Menu
```

```
Select an option ..... B      D = define new Search profile or
                                modify existing Search profile

                                B = define new Database profile or
                                modify existing Database profile

                                C = copy existing Search profile under
                                new name and modify

                                X = execute existing Search profile
```

```
Profile name ..... OEDB _____
  password ..... [           ] (existing profile only)
```

```
New profile name ..... _____ (Copy option only)
```

The next form for defining databases and output will already be filled out with the existing database profile configuration, as shown:

```
SUPERDEX/ENQUIRE : Define Database Profile Attributes

Database name ..... OEDB
password ..... [      ]

Profile password ... _____ (optional)

Delete ? ..... N      Y = delete this database profile
                          N = modify this database profile

Scope ..... _      G = all items in database
                          S = items in specific dataset(s)
```

Enter a valid database password in the space provided below the data base name, making sure that it grants access to all sets and items that are needed for the database profile. If modifying an existing database profile, use the same password that was specified when the database profile was defined. The password is not echoed to the screen for security reasons.

The Delete box is initialized to **N**, indicating that by default you are able to modify but not delete the database profile.

You may make any changes required to this form. To retain the current database profile password, do not modify the field; to remove the password, enter a backslash (\) in the first position in the password field. Press **ENTER** to proceed to the next form (and after completing each subsequent form). Once you have made the required modifications, you may save the search profile before finishing by pressing the **F7** function key. To flush all modifications and retain the original database profile, press **F8** to be returned to the Main Menu.

Deleting a database profile

An existing database profile may be deleted by specifying **B** at the Main Menu selection prompt and entering the name and password of the database profile.

On the next form, enter a valid password for the database in the space provided below the data base name. The password is not echoed to the screen for security reasons.

Enter **Y** in the Delete box to indicate that you want to delete (rather than modify) the current database profile. The database profile is deleted immediately.

SECTION 4:**Search profiles**



The Micro Interface formats are available as a separate option in ENQUIRE.

This section discusses *search profiles*, which are used for various custom database inquiries.

Search profiles perform specific pre-defined searches involving between one and 16 selection fields in up to 16 datasets in up to four databases. They may reference database profiles to determine input and output field attributes, although these attributes may be overridden as necessary within a search profile.

Chapter 1 Defining a new search profile

Function describes the procedures for *Defining a new search profile*, including specifying the base(s), set(s), and item(s) for selection and output.

Chapter 2 Modifying or deleting a search profile

Function reviews the methods of *Modifying or deleting a search profile*.

Chapter 3 Copying a search profile

Function shows the method for creating a new search profile based on an existing search profile by *Copying a search profile*.

Chapter 4 Executing a search profile

Function discusses the procedures for and options in *Executing a search profile*, including the various search criteria formats for accomplishing partial key, generic, range, and other types of searches.

Chapter 5 Reporting

Function the methods of *Reporting* entries located by use of a search profile are discussed, including printing and output in various HP3000 and microcomputer file formats.

CHAPTER 1:

Defining a new search profile

Main menu

To define a new search profile, run ENQUIRE with the **DEF** entry point, and then enter **D** in the Option box. Then enter a unique arbitrary name for the search profile of up to 14 characters, and leave the password blank. (If you enter the name of an existing search profile, ENQUIRE will automatically go into modify/delete mode rather than add mode.)

```
SUPERDEX/ENQUIRE : Main Definition Menu

Select an option ..... D   D = define new Search profile or
                             modify existing Search profile


                             B = define new Database profile or
                             modify existing Database profile

                             C = copy existing Search profile under
                             new name and modify

                             X = execute existing Search profile

Profile name ..... SHIPPING_____
  password ..... [           ] (existing profile only)

New profile name ..... _____ (Copy option only)
```

 **Do not use the name of an existing database as the search profile name unless you want to define a database profile containing the global attributes for a database--refer to the Defining a new database profile chapter in the Database profiles section for a discussion.**

Once both fields have been specified, press **ENTER** to process the screen's contents.

After completing the Main Menu, a form for defining the databases to access and the output format for the search profile is displayed. It may be filled in with:

- up to four databases to access
- a password for each database specified
- whether default input and output forms should be used or if custom forms have been prepared
- the format in which to output the data
- the optional search profile password
- whether to modify or delete the current profile

```

SUPERDEX/ENQUIRE : Define Database(s) and Global Attributes

Database name(s) ... _____
password(s) ... [ _____ ] [ _____ ] [ _____ ] [ _____ ]

Custom forms ? ..... N      Y = custom user-defined forms
                             N = standard pre-defined forms

Output format ..... _      _ = screen and printer only
                             1 = BINARY : binary format for HP3000
                             2 = SD      : self-describing format for HP3000
                             3 = ASCII   : comma/quote-delimited for PC wps
                             4 = WK1     : work file for Lotus 1-2-3 on PC
                             5 = DBF     : DBF file for dBASE on PC
                             6 = MAC     : tab-delimited for Apple Macintosh

Profile password ... _____ (optional)

Delete ? ..... N      Y = delete this search profile
                             N = modify this search profile

```


Defining databases and their passwords

Up to four databases may be specified for access in a search profile. Enter the unqualified name of each database (a file equation must already be set for databases that do not reside in the logon group and account). If less than four databases will be accessed, they must be left-justified.

The order in which the databases are specified is important, as databases are searched left-to-right, so each database must be logically dependent on a previously-defined database, with the leftmost database searched first, the next database second, etc.

A password for each database is entered below its name, and determines the user class that will be assigned to a user executing the search profile. Ensure that the password specified grants read access to all the datasets and items that are required for the search profile, as only those sets and items will be displayed and be available for selection (inaccessible sets and items are restricted and therefore not displayed on subsequent configuration forms).

If the required access to a database is obtainable only by using multiple user classes, the database may be specified more than once with a different password for each occurrence.

Passwords are not echoed to the screen for security reasons.

Forms specification

By default, ENQUIRE will generate standard VPLUS forms for entering and displaying data when the search profile is executed. The `Custom forms` box is therefore initialized to N.

The capability exists for using custom-defined VPLUS forms for input and output instead of using the default forms, in which case **Y** is entered. If so, the forms files must already exist--refer to the [*Customizing ENQUIRE*](#) section for information about creating custom forms.

Output format

ENQUIRE can report entries a user selects when executing the search profile in one or more of the following ways:

- display on the terminal screen
- list on a printer
- store to a binary-format file for use on the HP3000
- store to a self-describing (SD) file for use on the HP3000 and/or for conversion to a DIF file on a PC
- store to a file in comma/quote-delimited format for use by microcomputer programs, such as word processors
- store to a WK1-format file for use on a PC in Lotus 1-2-3
- store to a DBF-format file for use on a PC in dBASE
- store to a file in tab-delimited format for use on a Macintosh

By default, data may be output in screen and line printer format only, as specified by leaving the `Output format` box blank. The other options are referred to as *store* options, and one may be selected by indicating its corresponding number (**1-6**) in the `Output format` box.

- 1 BINARY** The data is written in the exact same format as it is stored in the database, useful for processing by HP3000 programs.
- 2 SD** The special **Self-Describing** format which is required by some programs on the HP3000.
- 3 ASCII** Fields are separated by a comma (,) and character strings are enclosed in double quotes ("). Entries are stored packed, with leading and trailing blanks removed from each field. This format is required by many PC programs.
- 4 WK1** Format for Lotus 1-2-3 worksheets (formerly called **WKS**). Contains a header line with item names or labels followed by the output table. The table itself has the range name **DATA** and can be read alone with the Lotus 1-2-3 command **\FCCRDATA**.
- 5 DBF** Format for dBASE III and dBASE III Plus database files. Headings (if defined) or item names are used as dBASE field names and truncated to 10 characters as required. All embedded special characters are converted to underscores. Subitems of compound **IMAGE** items are reassigned to discrete fields, with each field suffixed with the relative subitem number.
- 6 MAC** Popular format for various programs on the Macintosh. Each field is separated by a comma (,) and each entry is terminated by a **RETURN**.

Defining the search profile password

Access to the search profile may optionally be restricted by assigning an arbitrary password of up to eight characters. This password must be specified when executing the search profile, as well as when attempting to modify or delete the profile.

Editing and saving the current form

You may edit the form further by skipping forward from field to field with the **TAB** key or backward with the **SHIFT+TAB** keys in combination.

Once the form has been completed, press the **ENTER** key to save it in ENQUIRE's internal database. To flush the form without saving it, press the **F8** function key and you will return to the Main Menu.

Defining datasets to access

After the database/output form has been completed and saved, a separate dataset definition form is displayed for each database specified in the previous form, as identified in the upper right corner. Each form (one per database) defines:

- the datasets to access
- the order in which to access them

```
SUPERDEX/ENQUIRE : Define Dataset(s)                                base: OEDB
->Select up to 16 datasets and specify order by marking with a letter(A - Z)
_ M CUSTOMER                _ M ORDER-HEADERS                _ D ORDER-LINES
```

Each dataset is indicated by name and prefixed with a code defining the dataset type (**A**=automatic master, **M**=manual master, **D**=detail). Datasets are listed in the order in which they appear in the base's dataset list.

The datasets to be accessed in this search profile must be indicated, with one or more datasets per database, up to a maximum of 16 datasets for the search profile. Only the datasets that are required to satisfy the search requirements should be selected. SI-datasets (sets named "SI*n*") may not be selected.

Datasets are selected by entering an alphabetic character (**A-Z**) in the box to the left of each. Letters may be used to specify the order in which the datasets are to be accessed (**A** is first, **B** is second, etc.). The same letter may be assigned to multiple datasets, which causes ENQUIRE to sequence the datasets in the order shown on the form (the order of the database set list).

If more than one dataset in a database is used in a search profile, the sequence in which the datasets are accessed at execution time is important. ENQUIRE must start its search with a dataset upon which other datasets depend, either directly or indirectly. This dataset is referred to as the *primary* dataset, and all other datasets in the database (called *secondary* datasets) are dependent on the primary set or to a preceding secondary set. A dependence between two datasets is established by the presence of:

- an IMAGE path relating a master and detail dataset
- an IMAGE search field with the same item name as an item in another dataset
- an SI-path related to each set that references the same item in both sets

You must identify the primary dataset by specifying an **A** in the selection box. Secondary datasets are identified by other letters, where **B** must be dependent on **A**, **C** must be dependent on **A** or **B**, and so on. ENQUIRE is capable of determining the order of dependence once the primary dataset has been defined, so it is sufficient to identify the primary set (with an **A**) and the others with some other letter. In fact, ENQUIRE will change the order of dataset selection if dependence cannot be established within the sequence specified. Dependence must exist for all datasets specified.

Up to 48 datasets are displayed on one screen; additional sets are shown on one or more additional pages that are automatically displayed when **ENTER** is pressed after completing the current page. You may skip between the pages by using the **F2** and **F3** function keys.

Once all the datasets in a database have been displayed and the required ones selected, press **ENTER** to proceed to define the datasets for the next database (if multiple databases were defined on the database/output form). To instead flush the form without saving it, press the **F8** function key and you will be returned to the Main Menu. Other function keys may be used to perform different actions, as documented in the *Operation* section.

Defining dependency between datasets

If more than one dataset is used in a search profile, the dependence between datasets has to be established. This is automatically done based on identical item names in the datasets. If identical items do not exist or the default is not wanted, it can be defined via the ENQDEPS file (which can be file equated).

ENQDEPS.DEMO40.SUPERDEX is an example of how to define the linkages. The format for ENQDEPS is:

```
* COMMENT Lines can be entered using an asterisk (*) in column 1
*
* Define the linkage by defining the "TO" database information first,
* followed by the "FROM" database information.
*
* The format is:
*   "TO" database, followed by a semicolon (;)
*   "TO" dataset, followed by a period (.)
*   "TO" item, followed by an equal sign (=)
*   "FROM" database, followed by a semicolon (;)
*   "FROM" dataset, followed by a period (.)
*   "FROM" item
*
* For example:
*
DBASE2:DBASE2-DATASET.DSET-ITEM=DBASE1:DBASE1-DSET.DSET-ITEM
*
* This is a free format. Therefore it can contain spaces for
* readability
*
DBASE2 :DBASE2-DSET .DSET-ITEM = DBASE1 :DBASE1-DSET .DSET-ITEM
*
* This can also be used to change or define the linkage between 2
* datasets that are in the same database:
*
DBASE:DSET2.CUSTOMER-NUMBER = DBASE:DSET1.CUSTOMER-NO
*
* RULES:
* -All parameters are required.
* -The data types for the items MUST match. The value, exactly from
* the data record, is used to read the record(s) from the "TO"
* database.
* -The dataset names and items MUST be in uppercase.
```

Defining fields for selection

After you have completed and saved the dataset definition form(s), a separate field definition form is displayed for each dataset selected in the previous form. The base and set names are indicated in the upper right corner. Each form (one per dataset) defines:

- the fields to access
- the order in which to access them
- the type of access for fields that are used as IMAGE search fields or SUPERDEX SI-keys or SI-subkeys

Fields are listed in the order in which they appear in the dataset item list. Each field is indicated by its item name and immediately prefixed with C or N, which indicates in what format the data is represented, where:

- **C** = character (alphanumeric IMAGE data types U and X)
- **N** = numeric (IMAGE data types I, J, K, R, P and Z)

A leading prefix indicates the type(s) of key usage for fields that are used as IMAGE or SUPERDEX keys, where:

- **I** = IMAGE search field
- **S** = one SUPERDEX SI-key
- **IS** = IMAGE search field and one or more SUPERDEX SI-keys
- **SS** = multiple SUPERDEX SI-keys
- **s** = second, third or fourth SI-subkey in a concatenated SUPERDEX SI-key

```

SUPERDEX/ENQUIRE : Define Selection Item(s) base/set: OEDB /CUSTOMERS

->Select up to 16 items;specify order by marking first column with A-Z;
specify access by marking second column for IMAGE or SUPERDEX(I or S)

__ I N CUSTOMER-NUMBER __ C CUSTOMER-ABBR __ SS C CUSTOM-NAME
__ S C ADDRESS-1 __ C ADDRESS-2 __ S C CITY
__ C STATE __ N ZIP-CODE __ N PHONE-AREA-CO
__ N PHONE-PREFIX __ N PHONE-SUFFIX

```

For each dataset in a search profile, there is a maximum number of keys that can be selected, namely:

- one type "I" key and two type "S" keys, or
- one type "I" key and one type "s" key

Additionally, a type "s" key may be selected for the first dataset only.

The fields to be searched at execution time must be indicated, with one or more fields per dataset, up to a maximum of 16 items for the search profile. Only the fields that are required to satisfy the search requirements should be selected.

Fields are selected by specifying one or more letters in the box to the left of each item. In the first column of the box, enter an alphabetic character (**A-Z**) in the box. Letters may be used to specify the order in which the datasets are to be accessed (**A** is first, **B** is second, etc.). The same letter may be assigned to multiple datasets, which causes ENQUIRE to sequence the fields in the order shown on the form (the sequence of the dataset's item list).

For fields that are used as keys in IMAGE or SUPERDEX, indicate (in the second column of the box) the type of access desired by specifying **I** for IMAGE, **S** for SUPERDEX, or blank for non-key access. Fields that are defined as SUPERDEX SI-keys inherit their SUPERDEX attributes. For grouped SI-keys, only one SI-subkey defined in the SI-path needs to be a selection field. If neither **I** nor **S** is specified for any selection field for a dataset, the dataset is read sequentially.

The field that links the current dataset with the primary or other secondary set is always implicitly selected and may not be changed (nor does it count toward the search profile maximum of 16 items).

Up to 48 fields are displayed on a single page; if the dataset contains more than 48 fields, the other fields are displayed on one or more additional pages which are automatically displayed when **ENTER** is press after completing the current page. You may skip between the pages with the **F5** and **F6** function keys.

Once all the fields in a dataset have been displayed and the required ones selected, press **ENTER** to proceed to define the fields for the next dataset (if more than one dataset was previously selected). To instead flush the entire search profile without saving it, press the **F8** function key and you will be returned to the Main Menu. Other function keys may be used for various actions.

Defining SUPERDEX SI-path access

After you have completed and saved the selection item form(s), ENQUIRE checks which SUPERDEX SI-paths exist for each selection item. If it finds that there are multiple SI-paths configured for any chosen selection item (type "SS"), a separate SUPERDEX path definition form is displayed for each such selection item to specify which of multiple SI-paths to utilize

```

SUPERDEX/ENQUIRE : Define SUPERDEX path      base/set: OEDB/CUSTOMERS

-> Select one of the SUPERDEX SI-paths displayed by marking with an X :

      SI-path name          SI-subkey items
-----
_   CUSTOMER-NAME          : CUSTOMER-NAME
_   K CUSTOMER-NAME-KW    : CUSTOMER-NAME

```

The name and SI-subkeys of each SUPERDEX SI-path are displayed, prefixed by its type:

- K = keyworded
- G = grouped or super-grouped

Select the SI-path to use by marking it with an **X** in the box to the left of the desired SI-path.

Once the SUPERDEX SI-path has been chosen, press **ENTER** to proceed to define the selection fields for the next dataset (if it was previously selected). To instead flush the entire search profile without saving it, press the **F8** function key and you will be returned to the Main Menu.

Defining fields for output


After you have completed and saved the field definition form(s) or optional SUPERDEX path form, a separate output field definition form is displayed for each dataset in the search profile. The base and set names are indicated in the upper right corner. This form (one per dataset) defines:

- the fields to output
- the order in which to output them


```
SUPERDEX/ENQUIRE : Define Output Field(s) base/set:  OEDB/CUSTOMERS
->Select up to 128 fields and specify order by marking with letter(A-Z)

_ N CUSTOMER-NUMBER      _ C CUSTOMER-ABBR      _ C CUSTOMER-NAME
_ C ADDRESS-1            _ C ADDRESS-2          _ C CITY
_ C STATE                _ N ZIP-CODE           _ N PHONE-AREA-CO
_ N PHONE-PREFIX        _ N PHONE-SUFFIX
```

Fields are listed in the order in which they appear in the dataset item list. Each field is indicated by its item name and prefixed with C or N, which indicates whether the item is stored as Character (alphanumeric) or Numeric.

 **The fields to be outputted must be indicated, with zero or more fields per dataset, up to a maximum of 128 output fields for the search profile. For search profiles that access more than one dataset, it is required that at least one output field be specified for the last dataset defined but not for any other datasets.**

Fields are selected by entering a character in the box to the left of each field. Valid characters are **A-Z** and specify the order in which the fields are to be listed (**A** is first, **B** is second, etc.). You may want to reorder the fields for reporting requirements. To list the fields in the order in which they appear in the form, just assign the same letter to each.

Up to 48 fields are displayed on a single page. If the set contains more than 48 fields, the other fields are displayed on one or more additional pages which are automatically shown when **ENTER** is press after completing the current page. You may skip between the pages with the **F2** and **F3** function keys.

Once all the fields in a dataset have been displayed and the required ones selected, press **ENTER** to proceed to the fields for the next dataset, if defined. To instead flush the entire search profile without saving it, press the **F8** function key and you will be returned to the Main Menu. Other function keys have different actions.

Defining input and output field attributes

After you have completed and saved the output field definition form(s), a separate input/output field attribute definition form is displayed for each dataset in the search profile. The base and set names are indicated in the upper right corner. This form (one per dataset) defines:

- the prompt to display for each input field
- the heading to display for each output field
- edit mask specifications for alphanumeric and numeric fields
- the decimal point position for numeric fields

SUPERDEX/ENQUIRE : Define Labels & Formats base/set : OEDB/CUSTOMERS				
Item	Prompt	Heading	Edit spec	Decimal
N CUSTOMER-NO	_____	_____	_____	-
C CUSTOMER-NAME	_____	_____	_____	-
C ADDRESS-1	_____	_____	_____	-
C ADDRESS-2	_____	_____	_____	-
C CITY	_____	_____	_____	-
C STATE	_____	_____	_____	-
N ZIP-CODE	_____	_____	_____	-

Each output field selected in the previous form is indicated by its item name and listed in the order specified in the previous form.

If a database profile has been defined for any database in this search profile, its attributes are displayed on this form and any or all (or none) may be overridden as desired.

Refer to the *Defining a new database profile* chapter in the *Database profiles* section for information about defining prompts, headings, edit mask specifications, and decimal point position.

Completing the search profile

Up to 16 fields are displayed on a single page; if more than 16 fields have been selected, the other fields are displayed on one or more additional pages which are automatically displayed when **ENTER** is press after completing the current page. You may skip between the pages by using the **F2** and **F3** function keys.

Once all the fields in a dataset have been displayed and the required ones selected, press **ENTER** to proceed to define the fields for the next dataset (if it was previously specified). To instead flush the entire search profile without saving it, press the **F8** function key to return to the Main Menu.

CHAPTER 2:**Modifying or deleting a search profile****Modifying a search profile**

Existing search profiles may be modified in very much the same way and using the same forms as in defining a new search profile. Refer to the *Defining a new search profile* chapter for a discussion.

To modify an existing search profile, run ENQUIRE with the **DEF** entry point, then specify **D** in the Main Menu Option box and enter the name of the search profile and its password (if one was assigned), as shown:

```
SUPERDEX/ENQUIRE : Main Definition Menu
```

```
Select an option ..... D   D = define new Search profile or
                             modify existing Search profile

                             B = define new Database profile or
                             modify existing Database profile

                             C = copy existing Search profile under
                             new name and modify

                             X = execute existing Search profile
```

```
Profile name ..... SHIPPING_____
  password ..... [           ] (existing profile only)
```

```
New profile name ..... _____ (Copy option only)
```

The next form for defining databases and output will already be filled in with the existing search profile configuration, as shown:

```

SUPERDEX/ENQUIRE : Define Database(s) and Global Attributes

Database name(s) ... OEDB_  _  _  _
password(s) ... [    ] [    ] [    ] [    ]

Custom forms ? ..... N      Y = custom user-defined forms
                               N = standard pre-defined forms

Output format ..... _      _ = screen and printer only
                               1 = BINARY : binary format for HP3000
                               2 = SD      : self-describing format for HP3000
                               3 = ASCII   : comma/quote-delimited for PC wps
                               4 = WK1     : work file for Lotus 1-2-3 on PC
                               5 = DBF     : DBF file for dBASE on PC
                               6 = MAC     : tab-delimited for Apple Macintosh

Profile password ... _____ (optional)

Delete ? ..... N          Y = delete this search profile
                               N = modify this search profile
    
```

For each database, enter the same password in the spaces provided as when the search profile was defined. Database passwords are not echoed to the screen for security reasons.

The Delete box is initialized to N, indicating that the default mode is modification and not deletion.

You may make any changes required to this form. To retain the current search profile password, do not modify the field; to remove the password, enter a \ in the first position in the password field. Press **ENTER** to proceed to the next form, and after completing subsequent forms. To flush all modifications and retain the original search profile, press **F8** to be returned to the Main Menu. You may use other function keys for different results.

Each successive form is initialized with the current search profile configuration, unless

- a different database password is specified than that used when the search profile was defined
- different datasets are selected
- the order of the datasets is changed

in which case the existing search profile parameters are flushed and must be re-specified.

Deleting a search profile

An existing search profile may be deleted by specifying **D** at the Main Menu selection prompt and entering the name and password of the search profile, as shown.

The next form for defining databases and output will already be filled in with the existing search profile configuration. Enter a valid password for each database in the spaces provided. Passwords are not echoed to the screen for security reasons.

Enter **Y** in the Delete box to indicate that you want to delete (rather than modify) the current search profile; the search profile is deleted immediately.

CHAPTER 3:

Copying an existing search profile

A new search profile may be created based on an existing search profile by duplicating the search profile and modifying the copied version.

To copy an existing search profile, run ENQUIRE with the **DEF** entry point, then specify **C** in the Option box and enter the name of the existing search profile and its password (if one was assigned) and a new name under which the search profile should be duplicated, as shown:

```
SUPERDEX/ENQUIRE : Main Definition Menu

Select an option ..... C   D = define new Search profile or
                             modify existing Search profile

                             B = define new Database profile or
                             modify existing Database profile

                             C = copy existing Search profile under
                             new name and modify

                             X = execute existing Search profile

Profile name ..... SHIPPING_____
password ..... [      ]           (existing profile only)

New profile name ..... SHIPPING2   (Copy option only)
```

In this example, a new search profile **SHIPPING2** has been created as a duplicate of **SHIPPING**, and may now be modified.

The next form for defining databases and output will already be filled in with the existing search profile configuration, as shown:

```
SUPERDEX/ENQUIRE : Define Database(s) and Global Attributes

Database name(s) ... OEDB__  _____  _____  _____
password(s) ... [          ] [          ] [          ] [          ]

Custom forms ? ..... N      Y = custom user-defined forms
                             N = standard pre-defined forms

Output format ..... _      _ = screen and printer only
                             1 = BINARY : binary format for HP3000
                             2 = SD      : self-describing format for HP3000
                             3 = ASCII   : comma/quote-delimited for PC wps
                             4 = WK1     : work file for Lotus 1-2-3 on PC
                             5 = DBF     : DBF file for dBASE on PC
                             6 = MAC     : tab-delimited for Apple Macintosh

Profile password ... _____ (optional)

Delete ? ..... N          Y = delete this search profile
                             N = modify this search profile
```


Follow the steps for modifying an existing search profile as previously described.

CHAPTER 4:

Executing a search profile

Main menu

To execute an existing search profile, run ENQUIRE (with no entry point). Then enter the name of an existing search profile in the appropriate box, specify the search profile password (if one was assigned), and press **ENTER**. If you do not remember the name of the search profile you want to execute, specify a search profile of **?**, which will display all the search profiles that exist for a specified database.

 To execute the "?" profile, you must first setup a :FILE equation for DBENQ. For example:

:FILE DBENQ=DBENQ.SUPERDEX.SYS

You may alternately run ENQUIRE with the **DEF** entry point and specify **X** in the Option box to execute the search profile. This method is intended for development and not recommended for use by end users.

```
SUPERDEX/ENQUIRE : Main Execution Menu
```

```
Search profile name ..... SHIPPING_____
password ..... [          ]
```

Specifying values to search for

A prompt is issued for every field defined for selection. The label is either the *prompt* defined or, if none was specified, the item name:


```
SUPERDEX/ENQUIRE:  Input Selection Value(s)

-> Enter a selection value for each field :

Enter Shipper name  _____
Enter Shipment status _____

Entry limit: _____ Output file: _____
              _____ entries qualify
```

The specified values for each field are logically ANDed together, and ENQUIRE searches for all the entries that qualify based on a combination of all the values. To ignore a selection field (thereby qualifying all values), simply leave the selection value blank.

 **A field can be ORed with the previous field by beginning the second field with a comma (,), which is the OR symbol in ENQUIRE.**

Selection values are case-sensitive and are only upshifted for keyworded SI-paths, since the corresponding SI-keys are automatically upshifted and stored in upper case.

Each value entered for a Character (alphanumeric) item can be:

- an exactly-matching value
- a partial value (appended by an @)
- a generic value (containing embedded ?s)
- a greater-than, greater-than-or-equal-to, less-than, or less-or-equal-to construct (a value prefixed by >, >=, < or <=)
- a not-equal-to construct (a value prefixed by <>)
- a range of two values (separated by :)
- blanks, representing all values

For a Numeric item:

- an exactly-matching positive value (conditionally prefixed by a +)
- an exactly-matching negative value (prefixed by a -)
- a greater-than, greater-than-or-equal-to, less-than, or less-or-equal-to construct (a value prefixed by >, >=, < or <=)
- a not-equal-to construct (a value prefixed by <>)
- a range of two values (separated by :)
- blanks, representing all values

For items that are configured as SUPERDEX SI-keys, the following may additionally be specified:

- for grouped SI-paths, any value in any SI-key in the group
- for keyworded SI-paths, any significant word contained in the SI-key
- for all SI-paths, multiple values using Boolean operators



Alphanumeric values that contain any of the operators show above as valid characters must be enclosed in double quotes.

Searching for all values

To search for all values for a selection field, simply leave the field blank, which effectively ignores the field.

Alternatively, an @ may be specified in the selection field to qualify all entries. Although all entries are qualified using this method, it may be more efficient if the field is used as a SUPERDEX SI-key than to leave the field blank. Also, if the field is an SI-key, entries are returned in sorted order by this field.

For an alphanumeric or numeric field, blanks may be used to represent all values, and is useful for searches in which any value for a field is acceptable. This effectively ignores the selection field altogether. For example, you may have a search profile that includes an amount field but you only sometimes want to restrict the search based on an amount--on other occasions, entries with any amount should be included. For the latter case, simply specify a blank value.

Searching for partial values

ENQUIRE can search for entries using a partial value for alphanumeric items, appended with an @. The @ is treated as a wildcard (as in :LISTF) that represents any number of any character.

For example, to find all the entries that begin with "GENERAL," specify the value

GENERAL@

The partial value specified is compared with the entries, always starting with the first character so a construct like @ERAL is not allowed. The @ wildcard must always come at the end of the value. Characters after the @ are ignored.

Searching for generic values

The ? facilitates generic searches against alphanumeric items, and represents a single alphanumeric character (as in :LISTF). It may occur multiple times anywhere in the value, for example:

STR?NG

would locate "STRING," "STRONG," and "STRUNG."

The ? matchcode may also be used in combination with the @ wildcard, for example:

STR?NG@

would additionally find "STRINGER," "STRINGING," and "STRANGE."

The ? matchcode may also be used to locate entries in which the desired value does not begin in the first position; for example:

??RT?N

would locate "BARTON," "BURTON," "MARTIN," and "MORTON."



Instead of specifying a ? at the beginning of an argument, it is more efficient to define an offset when possible.

Searching for greater-than/less-than values

Greater-than and less-than searches, as well as greater-than-or-equal-to and less-than-or-equal-to searches, are accomplished for either alphanumeric or numeric items by using the **>**, **>=**, **<**, and **<=** operators to prefix the value. For example, to find all the entries greater than 1000:

>1000

To find all the entries that are greater than or equal to 1000:

>=1000

The **<** and **<=** operators work the same way for less-than and less-than-or-equal-to searches.

> and **<** operators are not available for keyword and grouped retrievals; use **>=** and **<=** instead.

Searching for not-equal-to values

Not-equal-to searches are accomplished for either alphanumeric or numeric items by using the **<>** operator to prefix the value. For example, to find all unpaid orders by testing a paid flag:

<>PD

When used in a search profile with multiple selection fields, any field value prefixed by the **<>** operator is AND NOTed with values for the other fields. For example, if the selection fields are CITY and STATE and **<>LOS ANGELES** is specified for CITY and **CA** is specified for STATE, all the entries in the state of California and not Los Angeles are selected.

Searching for a range of values

A range of values may be located in alphanumeric or numeric items by specifying the low and high endpoint values separated by a **:**. This searches for all the entries that contain values that occur between the specified values, inclusive. For example, to find all the entries that contain dates between April and June:

88/04/01:88/06/30

Ranges may alternatively be specified using the **>=** and **<=** operators in combination. For example,

>=88/04/01<=88/06/30

is equivalent to the previous example.

Searching for multiple values in a single field

Multiple values may be specified for a single selection field that is used as a SUPERDEX SI-key. This is done by including the following Boolean operators as delimiters:

- + for AND operation
- , for OR operation
- - for AND NOT operation


For example, to find all the customers in CALifornia, ORegon, and WASHINGTON:

CA,OR,WA

The search criteria are evaluated from left to right. If this does not impose the desired order of evaluation, parentheses may be used for grouping. For example, to find all the entries in a keyworded SI-key that contain both the words "COMB" and "BIND" or "HOLD" and "DRILL" but not any word beginning with "FASTEN":

(COMB+BIND) ,(HOLD+DRILL) -FASTEN

If the search profile utilizes more than one SI-key, parentheses may be imposed for only the selection value for the first SI-key.


 **Embedded spaces may not be included in relational search criteria, but >=, <= and : may be included.**

Searching for values that contain reserved characters

Several characters are reserved as operators in ENQUIRE, which may prevent entries with values that contain these characters from being located. For this reason, it is possible to instruct ENQUIRE to treat all characters literally in the search value, simply by enclosing the search value in double quotes, as shown:

"GUESS? JEANS"

would disable the treatment of ? as the matchcode operator.

 **If the field is a SUPERDEX key, the ? must also be disabled as a match character using an info string in SIMAINT, e.g. ;INFO="@ - "**

Searching for negative numeric values

Negative values in numeric items must be prefixed with a negative sign (-).

For numeric items of IMAGE data types P and Z, IMAGE distinguishes between numbers with no preceding sign and positive numbers. This distinction is only significant if the item is used as an IMAGE search field, which is a rare occurrence. If prompted for a value for a field that meets this condition, precede the value with a positive sign (+) and, if necessary, include a decimal point in the value. Though - and + are reserved operators in ENQUIRE, they do not have to be enclosed in double quotes when used as a sign of a numeric value.

Searching compound IMAGE items

Only the first subitem in a compound IMAGE item (commonly referred to as an arrayed or tabled item) is searched. If the specified value occurs in any subitem but the first, the entry may not qualify. (All subitems, however, are displayed on output.)

Specifying limit on number of entries to return

For testing or other purposes, you may impose a limit on the number of entries to be located and returned. Specify the number of entries to limit the search to in the Entry limit box, or leave it blank to keep it unlimited.

Specifying file for output

The `Output file` field outputs data to a file in a pre-defined format. The output file is created with 32 extents and 4 allocated, thereby reserving ample space for entries from subsequent searches to be appended. If the file's *flimit*, *numextents*, *maxextents*, or other attributes are insufficient, they may be overridden with a file equation in the field.

If you need to store the data entries to a file, enter the file name, with or without a group name. If you want to append the entries to an existing output file, suffix the file name with `/A`.

If output will be printed (**F4** function key) and the printer device is other than **LP**, the device of the desired printer may be specified in the Output file field as either:

```
;DEV=LASER
```

to direct printed output to device class **LASER**, or

```
=*LASER
```

if a global file equation had been issued for **LASER**.

The output can be redirected to a file by specifying:

```
=filename ;DEV=DISC;SAVE;NOCCTL
```

CHAPTER 5:

Reporting

Reporting the number of qualifying entries

Once the selection values have been specified, press **ENTER** to have ENQUIRE proceed with the search. If the search requires a serial read, a message is displayed requiring that the serial read be confirmed by pressing **F1 - F7** or canceled by pressing **F8**.

Once the search is completed, the number of qualifying entries is displayed in the lower left area of the screen.

Often, just knowing the qualifying number of entries is sufficient and it is not necessary to view the entries, in which case the user may proceed to another inquiry by specifying new values or go back to the Main Menu by pressing **F8**.

Otherwise, ENQUIRE can output the entries found in several ways:

- display entries on the terminal screen (press **F2**)
- display only totals and common values on the screen (press **F3**)
- print to a line printer (press **F4**)
- store to a file in the pre-defined store format (press **F5**)

Reporting entries to the screen (LIST ALL)

Entries are shown one at a time on the terminal screen by pressing the **F2** function key. Only the output fields defined are shown, with their configured *headings* and in the format determined by their configured *edit masks*.

For fields that contain compound IMAGE items, each subitem value is displayed on its own line, with the subitem number appended to the heading.

Entries are displayed in either a vertical or horizontal format, depending on their overall length. Entries that can each fit on a single line are displayed in horizontal format, up to 18 entries per page. Those that cannot fit on a single line are displayed one per page, up to 16 fields of an entry per page.

Entries that can fit on one line are displayed in the following format:

SUPERDEX/ENQUIRE : Display Qualifying Entries				
Customer-#	Order-#	Part-#	Invoice- line-no	Unit-price
2100304	701193	A626765N	17	2695
2100304	701193	Y4403CR	16	169
2100304	701193	R9530609	15	275
2100304	701193	SRA	14	68
2100304	701193	BCMRC21BE	13	1176
2100304	701193	G27-12	12	2961
2100304	701193	C15-BLK	11	422
2100304	701193	BCMRC21BK	10	1176
2100304	701193	WES40290	9	601
2100304	701193	710-01	8	156
2100304	701193	482-2	7	123
2100304	701193	332-01-RED-M	6	9
2100304	701193	331-01-GRN-M	5	9
2100304	701193	SCM1312	3	382
2100304	701193	CLI-PC	2	102
2100304	719117	A615724	1	1960

Up to 18 entries are displayed on a single screen. You may page ahead to display additional entries by pressing the **F2** function key.

If all of the fields cannot fit on a single line, the following vertical display format is used instead:

```
SUPERDEX/ENQUIRE : Display Qualifying Entries

Field names and contents :

Shipper number      18
Shipper's name      BUFFALO AND ERIE CO PUB LIBRARY
Address             LAFAYETTE SQ OAK ST ENT
                   BUSINESS OFFICE
City                BUFFALO
State               NEW YORK
Zip code            14203
Shipment status     0
Customer Number     302503
Customer name       COUNTY OF ERIE
Address             95 FRANKLIN STREET
City                BUFFALO
State               NY
Zip code            14202
Balance             $8321.91
Amount              $10.00
```

Up to 160 fields can be displayed for any entry, but only 16 lines are displayed on the screen at a time. You may skip between multiple pages with the **F5** and **F6** function keys.

Additional entries may be displayed by pressing the **F3** function key; previous entries are displayed with **F2**.

In either format, you may proceed to another inquiry by pressing the **F7** function key or go back to the Main Menu by pressing **F8**.

Reporting totals only to the screen (LIST SUMS)

Rather than displaying all the entries on multiple pages, a single page may be displayed showing only the totals of the entries found by pressing the **F3** function key. Alphanumeric values are not totaled; numeric values are totaled only if **S** was specified as first character in the edit mask.

Total fields are automatically extended by two character places to accommodate larger display values. Alphanumeric values and numeric values which are not totaled are displayed only if their values are the same for all the entries; otherwise, the field is filled with asterisks (*):

SUPERDEX/ENQUIRE : Display Qualifying Entries				
Customer-#	Order-#	Part-#	Invoice- line-no	Unit-price
2100304	701193	*****	*****	10529+
2100304	719117	A6175724	1	1960+
2100304	728090	A626765N	1	2695+
2100304	815402	*****	*****	15477+
2100304	831252	*****	*****	118+
2100304	909375	*****	*****	17563+
2100304	916401	568-01	1	159+
2100304	928311	*****	*****	16008+
2100304	928312	*****	*****	11484+

If the search involved multiple datasets, only the numeric values of the last dataset accessed are summed and displayed.

Printing entries on printer (PRINT)

The entries are printed on the printer by pressing the **F4** function key. Output is by default sent to the system line printer (device class **LP**), but may be redirected to another printer by a file equation for **ENQLIST** or specifying the printer device in the Output file field, as described previously.

Entries are printed one per line, with a two-line heading. If the printer line width is not sufficient to display the entire entry, the entry is split and displayed on two lines. In this case, a four-line heading is printed.

If two lines are not sufficient to display the entire entry, entries are printed piecewise: They are printed one per line and truncated at the end of the printer line. Then the continuation of all entries is printed, starting from the item where the split was made; and this is repeated until the entries are completely printed. In every continuation line, the first item of the entry - usually a key - is repeated, thus corresponding lines can be identified by the first field.

Printer line width (default: 132) and number of lines per page (default: 66) can be configured via JCWs:

:SETJCW ENQLPWIDTH=nn

defines printer line width, the maximum allowed is 132

:SETJCW ENQLPLINES=nn

defines lines per page.

The JCWs can be set before calling ENQUIRE, or in BREAK.

Storing entries to a file (STORE)

The entries are written to the file specified in the Output file box by pressing the **F5** function key.

The output file may be created on the HP3000 or, if you are using Reflection, automatically downloaded to your microcomputer. This decision is made by ENQUIRE based on the specified file name. The output file will be created on the microcomputer rather than the HP3000 if the file name meets one of the following criteria:

- first character is a backslash (\)
- first character is a period (.)
- second character is a colon (:)

Refer to the *Reflection interface* chapter of the *Batch and micro interfaces* section for more information.

If the specified file is built on the HP3000 and it already exists, and the append option (file name suffixed with /A) was not specified, a message is displayed and a different file name may be entered, or the existing file may be overwritten by pressing **F5** again.

The format in which the entries are stored is pre-defined in the search profile as either ASCII (comma/quote delimited, BINARY (HP3000), DBF (dBase), SD (self-describing), WK1 (Lotus), or MAC (Macintosh).

SECTION 5:

Batch and micro interfaces

This section describes ENQUIRE's batch and microcomputer interfaces.

Chapter 1 Batch Interface

Function describes the *Batch interface*, which is useful for incorporating advanced retrieval capabilities to application programs by allowing ENQUIRE to be called directly.

Chapter 2 Micro Interface

Function describes the *Micro interface*, which permits files to be output in various formats for use on microcomputers. It discusses manually and automatically downloading files to microcomputers and gives an example of a Reflection command file.

Chapter 3 Lotus Interface

Function describes and gives an example of ENQUIRE's *Lotus interface*, which permits an ENQUIRE search profile to be executed and the results utilized from within Lotus 1-2-3 without exiting.

Chapter 4 Controlled Interface

Function describes the *Controlled interface*, which is useful for incorporating ENQUIRE into master programs like TaskMaster.

CHAPTER 1:

Batch interface

ENQUIRE can execute search profiles in batch by providing responses directly in the job stream or by utilizing separate input files.

This same batch facility is also useful for executing search profiles from within programs and other systems, such as Reflection command files.

Creating an input file

For batch and program operation, an input file may first be created with the search profile execution parameters. The input file may be built using any editor, with one parameter per line, in the following format:

<i>line</i>	<i>description</i>
1	search profile name (maximum 14 characters), followed by / and the search profile password (maximum 8 characters)
2 - <i>n</i>	selection values (one per line)
<i>n</i> +1	search limit (or blank if no limit imposed)
<i>n</i> +2	output file name (file name + optional group name + optional /A)

The following shows a sample input file called OVER90SP being created in EDIT/3000:

```
:EDITOR
HP32201A.07.17 EDIT/3000
(C) HEWLETT-PACKARD CO. 1985
/A
 1  PASTDUEOVER90/password
 2  CA
 3  >90
 4  1000:10000
 5
 6  OVER90.DATA/A
 7  //
/K OVER90SP
/EXIT

END OF SUBSYSTEM
```

This search profile execution locates customers in California who are more than 90 days past due and owe between 1,000 and 10,000 dollars. The search profile is called PASTDUEOVER90, three search criteria are specified (note how the range is broken over two lines), no search limit is imposed, and the entries found are appended to the existing output file OVER90.DATA.

Executing a search profile in batch mode

ENQUIRE may be run in batch by including parameters, one per line, in the same format as the input file, as shown:

```
!JOB OVER90,MGR.AP
!RUN ENQUIRE.PUB.SUPERDEX
PASTDUEOVER90/password
CA
>90
1000: 10000

OVER90.DATA/A
!EOJ
```

Alternatively, the search parameters may be specified in an input file and run from a job stream simply by adding a single command that runs ENQUIRE with the **BATCH** entry point, redirects its \$STDIN file to the input file, and redirects its \$STDLIST to \$NULL, as shown:

```
:RUN ENQUIRE.PUB.SUPERDEX,BATCH;STDIN=inputfile;STDLIST=$NULL
```

This causes ENQUIRE to execute the search profile and parameters defined in the inputfile and to suppress all its output with the exception of the output data file created.

Executing a search profile from another program

This same technique may be used to execute search profiles from within your own programs. Create the input file and run ENQUIRE as a son process of your program (using the CREATEPROCESS intrinsic) with the BATCH entry point and its \$STDIN and \$STDLIST redirected, as shown above.

CHAPTER 2:

Micro interface



The microcomputer interface is available as an add-on option to the standard SUPERDEX/ENQUIRE package.

Automatically downloading store files to the microcomputer

The output file is automatically built on the microcomputer if you are using Reflection and the file name specified in the Output file field meets one of the following criteria:

- first character is a backslash (\)
- first character is a period (.)
- second character is a colon (:)

Make sure to include the correct extension (.PRN, .WK1, or .DBF) on the file name.

Manually downloading store files to microcomputer

Files that have been created by ENQUIRE's store option may be manually transferred to a PC, Macintosh[™], or other microcomputer by various programs, such as Reflection[™], AdvanceLink[™], or Business Session[™]. Since these products are normally also used to emulate block mode, their file transfer facilities should be readily available.

In transferring files, make sure the transfer is done in **binary** format. If the micro file extension is not specified, the correct extension for the specified output file format is automatically appended.

Executing a search profile from a Reflection command file

The batch facility is also useful for executing a search profile from within a Reflection command file invoked from a PC.

The following Reflection command file contains embedded execution parameters, rather than having them contained in a separate input file. This example extracts entries from an IMAGE database, writes them to a file in dBASE format, and downloads the file to a microcomputer:

```
BACKGROUND
IF V2 = ""
  LET V3 = "MAILLIST.DBF"
ELSE
  LET V3 = V2 & ".DBF"
ENDIF
PTRANSMIT "RUN ENQUIRE.PUB.SUPERDEX,BATCH;STDLIST=$NULL"
PTRANSMIT "CUSTDB"
PTRANSMIT V1
PTRANSMIT ""
TRANSMIT "ENQTEMP^M"
WAIT FOR "^Q"
RECEIVE V3 FROM ENQTEMP BINARY DELETE
PTRANSMIT "PURGE ENQTEMP"
```

In this example, the CUSTDB database is searched for a specified value (V1), and the qualifying entries are written to the file ENQTEMP. This file is then transferred to the microcomputer under the name MAILLIST.DBF or a specified file name (V2.DBF), deleting any local file with the same name, and then the ENQTEMP file on the HP3000 is purged.

CHAPTER 3:

Lotus interface

Using ENQUIRE's batch facility in conjunction with a file transfer utility, it is possible to invoke an ENQUIRE search profile from within Lotus 1-2-3 using values specified in a worksheet and then access the resulting entries from within Lotus, without the need to exit.

The following Reflection command file (ASCII file that you create on your PC which can be named anything you wish) is executed in background on the PC and waits for the file **SDX.PRN** which contains the ENQUIRE batch commands and selection values to be created (be downloaded):

```
BACKGROUND
:START
IF EXIST("SDX.PRN")
  SEND SDX.PRN TO ENQIN ASCII DELETE
  ERASE SDX.PRN
  PTRANSMIT "FILE DBENQ=DBENQ.DEMO.SUPERDEX"
  PTRANSMIT "RUN ENQUIRE.PUB.SUPERDEX,BATCH;STDIN=ENQIN"
  RECEIVE ENQ.WK1 FROM ENQTEMP BINARY DELETE
  PTRANSMIT "PURGE ENQIN"
  PTRANSMIT "PURGE ENQTEMP"
ENDIF
WAIT 0:0:5
GOTO START
```

To execute this command file within Reflections, press the COMMAND LINE function key and enter the name of the file. Once this file is running, switch over to Lotus (but do not end the Reflection session).

The Lotus worksheet should contain a macro which generates the file **SDX.PRN**

```
\E          {WINDOWSOFF}
            {OPEN SDX.PRN,W}
            {WRITELN "SIMPLE"}          (Enquire search profile name)
            {WRITENL A2}                (Spreadsheet cell containing search arugment)
            {WRITELN ""}
            {WRITELN "ENQTEMP"}
            {CLOSE}
```

This macro generates the ENQUIRE batch commands using the contents of cell A2. This macro has to be adapted to the worksheet and to the search profile.

When this macro is executed, the file **SDX.PRN** is generated. The background command file transfers this file to the HP3000, executes ENQUIRE and transfers the result WK1 file back to the PC. All this occurs in background, while the user can continue to work on his worksheet. The new WK1 file can be accessed by the **/FILE RETRIEVE** or **/FILE COMBINE** commands.



It is important for the SDX.PRN file to be created in the same directory that the Reflection command file is expecting it!

When using the **/FILE RETRIEVE** command, the new worksheet has correct columns width and cell formats (including decimal point) as well as column heading. The column heading can be suppressed by **/FILE COMBINE** command using the range **DATA**.

CHAPTER 4:

Controlled Interface

ENQUIRE can execute a search profile in line mode instead of using VPLUS screens. This is done by using the **CONTROLLED** entry point.

```
:RUN ENQUIRE.PUB.SUPERDEX,CONTROLLED
```

In this mode ENQUIRE will prompt for the search profile, optionally for the password, for all selection values, for the limit, and for the output file name. If an error is detected, the corresponding message is displayed and a reprompt occurs. After the search profile is executed, ENQUIRE prompts for the next action, i.e. next selection, next profile, or exit. At any time, *//* can be entered as a response, which acts like **F8** in VPLUS mode.

```
ENQUIRE VERSION 3.1 COPYRIGHT DR. MATT/IABG (1988,1989)

DEMO VALID UNTIL SAT, AUG 11, 1990

Enter search profile >
SIMPLE
Enter customer name >
U@
Enter limit >

Enter output file >
SIMPLOUT
15 entries retrieved
next action: S(election) / P(rofile) / E(xit) >
E
```

Though it is possible to run ENQUIRE with the CONTROLLED entry point directly as a session, it is intended to be run either from a REFLECTION command file, a PC program using REFLECTION's Application Interface or as son process under control of a master process. Such a master process can be Office Extend TaskMaster.

TaskMaster's job is to bring the HP3000 data directly into PC applications. To retrieve the data from within Lotus, the user simply brings up the 1-2-3 command line and performs a /FileRetrieve specifying an Office Extend HostDisk "mapped-task" file entry on the HP 3000 (X:REPORT32.WK1, for example).

This action causes Office Extend to begin TaskMaster operation with a specified task script, in this case one designed to control ENQUIRE. TaskMaster gets any input required from the PC operator and displays messages on the PC screen. ENQUIRE extracts the requested data and TaskMaster returns the resulting file directly to Lotus.

SECTION 6:

Customizing ENQUIRE

This section describes the facilities for customizing ENQUIRE to suit your environment.

Chapter 1 Custom forms

Function discusses the criteria for setting up *Custom forms* for input and output rather than using ENQUIRE's default VPLUS forms.

Chapter 2 Native language support

Function reviews ENQUIRE's handling of *Native language support* and includes instructions for configuring the native language and modifying ENQUIRE's message catalog.

Chapter 3 HPFD support

Function shows how to display Business Basic's *Floating Point Decimals*.

CHAPTER 1:

Custom forms

Custom forms

By default, ENQUIRE generates standard VPLUS forms for data input and output.

Alternatively, you may define custom forms (in FORMSPEC) and store them in the same forms file as the default standard forms. The standard forms file is FOENQ nnn .SUPERDEX.SYS, where nnn is the *langid* (000 is Native-3000).

If defining custom forms for a search profile, both custom input and output forms must be defined. For either or both input and output forms, a custom help form may be defined.

Custom menus

ENQUIRE has a "MENU" entry point, which allows you to define a custom menu screen. This screen is used to display search profile choices for a user and then for the user to select the profile without knowing the name of the profile.

When the MENU entry point is used, ENQUIRE will display the **MENU** form. Optionally, a custom help form can be defined using the same format as ENQUIRE's standard custom help, **MENU**_____H (15 characters).

The **MENU** form contains a one-character selection field for each search profile. The name of the field must be the exact name of the search profile to be executed with that selection. For example, to choose the CUST-NAME search profile the **MENU** field for this choice must be CUST-NAME also.

The user then selects a search profile by entering a character (i.e. X) in the one-character selection field. Using the **MENU** form the user does not need to remember, or know, the names of the available search profiles.

Additionally, if search profile passwords are utilized, the **MENU** form can contain an optional PASSWORD field. This field should be defined as a non-echo, eight-character field. If this field is entered, the value is used as the password for the selected search profile.

For an example of a **MENU** VIEW form, see the **MENU** form in ENQFO000.DEMO40.SUPERDEX.

Custom input forms

The input form name must be comprised of the letter **S** followed by the search profile name. For example, the input form for the SHIP search profile would be named *SSHIP*.

The names of each input field (in which selection values are entered) must be the same as the first 15 characters of the corresponding item name (the last character is truncated). For item names that include special characters, replace each special character with an underline () when specifying the field names, since VPLUS disallows all other special characters. The sequence of the fields, as well as their positions on the form, is arbitrary.

Additionally, three fields may be included in the form, described below with their FORMSPEC attributes:

<i>name</i>	<i>description</i>	<i>f</i> type	<i>d</i> type
LIMIT	search field limit (optional input field)	O	DIG
ENQFILE	output file name for store option (input field, required only if an output format has been specified)	O	CHAR
ENTRIES	number of qualifying entries found (required output field)	D	DIG

The function keys operate in the same manner as in the default input form, specifically:

Default Input Form Function Keys		
Key	Label	Description
F1	HELP	Displays corresponding (context-sensitive) help form
F2	LIST ALL	List the qualified entries
F3	LIST SUM	List the totals of the qualified entries
F4	PRINT	Print the qualified entries to the printer
F5	STORE	Write the qualified entries to the designated file
F6		
F7		
F8	MENU	Go back to Main Menu, flush changes

Custom output forms

The output form name must be comprised of the letter **R** followed by the search profile name. For example, the output form for the SHIP search profile would be named *RSHIP*.

The names of each output field (in which data entry values are displayed) must be the same as the first 15 characters of the corresponding item name (the last character is truncated). For item names that include special characters, replace each special character with an underline () when specifying the field names, since VPLUS disallows all other special characters. The sequence of the fields, as well as their positions on the form, is arbitrary. All the desired fields in a dataset must be displayed on one form.

ENQUIRE can display up to 16 separate entries on a single page. To configure such a form, specify the corresponding item names on the first line of the form and unique, arbitrary names on subsequent lines. It is recommended that the first 14 characters of the item name suffixed by the numbers from 1 through 16 be used.

For fields that consist of compound IMAGE items, one display field should be provided for each subitem. The name of the first field must be the same as that of the item (which cannot exceed 15 characters), with each additional field having an arbitrary name but immediately following the first subitem field and each other. It is recommended to use the same item name suffixed by the subitem number; for example, the fields for the 4I2 field called QUARTER-TOTAL would be QUARTER-TOTAL, QUARTER-TOTAL2, QUARTER-TOTAL3, and QUARTER-TOTAL4.

The function keys operate in the same manner as in the default output form, specifically:

Default Output Form Function Keys		
Key	Label	Description
F1	HELP	Displays corresponding (context-sensitive) help form
F2	PREVIOUS ENTRIES	Displays previous page of entries
F3	NEXT ENTRIES	Displays the next screen of entries
F4		
F5		
F6		
F7	SELECT/ OUTPUT	Return to the Input /Selection screen
F8	MENU	Go back to Main Menu, flush changes

Custom help forms

A custom help form may be defined for either or both the custom input and output form, which will be accessible via the **F1** function key from each form. The content of the help form is completely freeform.

The help form name must be comprised of the **S** (for input) or **R** (for output) prefix character, followed by the search profile name, followed by a variable number of underlines () and the letter **H**, such that the total length of the form name is 15 characters. For example, the custom help form for the input form of the SHIP search profile would be named *SSHIP_____H* (9 underline characters).

CHAPTER 2:**Native language support**

Setting the native language

ENQUIRE can either be installed in a particular native language, or the language can be defined when the program is run.

To install ENQUIRE in a native language, patch the program with the octal value of the desired language ID number (*langid*) preceded by a zero, as shown:

```
:RUN PATCH.PUB.SYS  
  
FILE=?ENQUIRE.SUPERDEX.SYS  
?MG,11  
000000,010  
?EXIT
```

In this example, the native language is changed from Native-3000 (000) to German (octal 010, decimal 008).

To define the native language at run time instead, or to override the installed native language, indicate the one or two rightmost significant digit(s) of the *decimal langid* (*nn*) as a parameter on the :RUN command, as shown:

```
:RUN ENQUIRE.SUPERDEX.SYS;PARM=nn
```

The languages supported on the HP3000 with their language ID numbers in decimal and octal are as follows:

<i>Language</i>	<i>Decimal langid</i>	<i>Octal langid</i>
Native-3000	000	000
American	001	001
Canadian-French	002	002
Danish	003	003
Dutch	004	004
English	005	005
Finnish	006	006
French	007	007
German	008	010
Italian	009	011
Norwegian	010	012
Portuguese	011	013
Spanish	012	014
Swedish	013	015
Katakana	041	051

Forms file (with Native Language Support)

The forms files that contain standard and custom input and output VPLUS forms in various languages are FOENQ nnn .SUPERDEX.SYS, where nnn is the language ID number (*langid*). Forms files are provided for several languages, and additional forms files may be created for languages not provided.

The forms files are provided in source (VFORM) format. It is recommended that they be compiled into the fast (VFAST) format using FORMSPEC to increase execution speed.

Error and status message catalog (with Native Language Support)

Compiled message catalogs that contain error and status messages are ERENQ nnn .SUPERDEX.SYS, where nnn is the language ID number (Native-3000 is 000).

Sources are provided for each language in the files CAT nnn .SUPERDEX.SYS, and may be modified in any editor and recompiled using GENCAT. Refer to HP's **Native Language Support Reference Manual** for further information.

A table of ENQUIRE's error and status messages, along with their meanings and recommended actions, is shown in the *Error and exceptional conditions* appendix.

CHAPTER 3:

HPFD support

To instruct ENQUIRE to display K2 or K4 as floating point decimals enter the command

SETJCW SIHPFD=1

before starting ENQUIRE.

APPENDIX A:

Database structural changes

Certain changes to the structure of databases used in database and search profiles result in the profiles being unusable. This is because sets and items are internally identified by number in the **DBENQ** database, and any database structural modification that causes set or item numbers to change invalidates the corresponding profiles.

Set names and numbers are tracked within the DBENQ database and allow ENQUIRE to detect that a structural change involving datasets or items has occurred. In this case, an appropriate message that the profile is invalid is issued.

ENQUIRE is capable of updating set and item numbers within the DBENQ database when new sets and items are added and existing sets and items are deleted, as well as updating set and item names when sets and items are renamed.

Two options exist as entry points to ENQUIRE which update database and search profiles following database structural changes:

- **STRUCT** option, for a structural change that does not involve renames of either sets or items
- **RENAME** option, for a structural change that involves only set or item renames

Following a database structural change, ENQUIRE must be run with the appropriate option and the affected database identified. To do so, log on as the creator of the affected database and run ENQUIRE as follows:

```
:RUN ENQUIRE.SUPERDEX.SYS,STRUCT;INFO="base"
```

or

```
:RUN ENQUIRE.SUPERDEX.SYS,RENAME;INFO="base"
```

Both options cause ENQUIRE to search the DBENQ database for any database profile and all search profiles that access the specified database and automatically update them to reflect the new database structure.

If multiple DBENQ bases are used, ENQUIRE must be run separately against each DBENQ base that contains a database profile or search profiles relating to the affected base. This may be done by issuing the necessary file equation for each DBENQ base.

APPENDIX B:**Internal structures**

The DBENQ database

Search profiles are maintained in the **DBENQ** database, which by default resides in SUPERDEX.SYS and which may be moved to or duplicated in another group/account.

It is necessary to assure that the capacities of the datasets are sufficient to contain all the entries required by the search profiles. (DBGENERAL customers should configure the DBENQ database in the Automatic Capacity Manager using option 3.1 and use option 3.2 to maintain the capacities.)

The datasets and their contents are as follows:

- | | |
|-----------------|---|
| ENQ | Master dataset; contains one entry per database profile and one entry per search profile. |
| ENQR | Detail set; contains one entry per secondary dataset (primary dataset information is stored in the ENQ set). |
| ENQF | Detail set; contains one entry for the attributes of each field, including fields defined in database profiles. |
| SETINFO | Master set; contains one entry per accessible dataset in each database used in any search profile (including those datasets that are not used in any search profile). |
| ITEMINFO | Master set; contains one entry per accessible item in each database used in any search profile (including those datasets that are not used in any search profile). |

Determining ENQUIRE access method

ENQUIRE establishes the optimum method for accessing data entries based on existing IMAGE paths and SI-paths. The access method is determined when the search profile is defined, and is done in the following sequence:

1. If both an IMAGE path and SUPERDEX SI-key (type "I" and "S" keys) have been specified for a single field, ENQUIRE will use the IMAGE search field unless the value is blank, in which case the SUPERDEX SI-path will be used.
 2. If a value other than a full key value is specified for an IMAGE key, a sequential read is performed, as IMAGE does not support partial-key retrieval.
-

3. If a single SUPERDEX SI-key is used as a selection value, ENQUIRE performs indexed access.
4. If multiple SUPERDEX SI-keys are included as search fields and neither is blank, ENQUIRE performs relational access. If one value or the other is blank, indexed access is used.
5. If the search value includes Boolean operators, ENQUIRE performs a relational access.
6. If no IMAGE search field or SUPERDEX SI-key is included as a search field, ENQUIRE performs a sequential read.
7. If no value is specified for a selection field, the field is effectively ignored.

At execution time, for searches that involve multiple datasets, ENQUIRE reads the qualifying entries in the primary dataset and records the value(s) of the field(s) used to logically link to secondary sets. These values are then used to locate matching entries in any secondary sets. If no secondary entries exist, the primary entry is disqualified, since an AND condition must be satisfied.

APPENDIX C:**Maximum limits**

The following table identifies ENQUIRE's internal limits. Most limits are not checked, and results when exceeded are unpredictable.

ENQUIRE maximum limits

<i>Facility</i>	<i>Maximum limit</i>
Databases per search profile	4
Datasets per search profile	16
Selection fields per search profile (excluding IMAGE search field)	16
Output fields per search profile	128
Number of IMAGE search fields per dataset per search profile	1
Number of SUPERDEX SI-keys	2
Number of SUPERDEX SI-keys in which selection field is second SI-subkey	1

APPENDIX D:**Error and exceptional conditions****ENQUIRE error, exceptional, and status messages**

The **ENQUIRE error, exceptional, and status messages** table lists the various ENQUIRE error conditions, exceptional conditions, and status messages that could be issued by the ENQUIRE program, their meanings, and their corrective actions.

Message catalogs are supplied in various native languages, and may be modified as desired. Refer to the *Customizing ENQUIRE* section for more information.

ENQUIRE error, exceptional, and status messages

<i>Type</i>	<i>Message number / description</i>
<i>Message</i>	1 NO MESSAGE DEFINED
<i>Meaning</i>	No message is defined for this error.
<i>Action</i>	Call Bradmark Technical Support if assistance is required.
<i>Message</i>	2 DATABASE CANNOT BE OPENED <i>(followed by reason)</i>
<i>Meaning</i>	ENQUIRE is unable to open the specified database due to the error displayed, typically because an invalid password was specified or no file equation was set for a database residing outside of the logon group/account.
<i>Action</i>	Correct the cause of the error and retry.
<i>Message</i>	3 NO PREVIOUS PAGE
<i>Meaning</i>	You are currently accessing the first (or only) page of this form.
<i>Action</i>	Note the condition and continue.
<i>Message</i>	4 NO NEXT PAGE
<i>Meaning</i>	You are currently accessing the last (or only) page of this form.
<i>Action</i>	Note the condition and continue.
<i>Message</i>	5 TOO MANY FIELDS SELECTED
<i>Meaning</i>	More than 16 input fields or 128 output fields were selected.
<i>Action</i>	De-select (with a SPACE) one or more fields until the maximum limit is no longer exceeded.
<i>Message</i>	6 NO FIELDS SELECTED, AT LEAST ONE MUST BE
<i>Meaning</i>	At least one field must be selected for each dataset.
<i>Action</i>	Select one or more fields.
<i>Message</i>	7 SEARCH PROFILE DOES NOT EXIST
<i>Meaning</i>	You are attempting to execute a nonexistent search profile.
<i>Action</i>	Retry with a valid search profile (or specify ? to display all configured databases and search profiles).

ENQUIRE error, exceptional, and status messages (cont'd)

<i>Type</i>	<i>Message number/description</i>
<i>Message</i>	8 INCONSISTENCY; PLEASE REDEFINE PROFILE
<i>Meaning</i>	An inconsistency has been detected between the search profile and one or more of the databases it accesses .
<i>Action</i>	Refer to the <i>Database structural changes</i> appendix for a discussion.
<i>Message</i>	9 CUSTOM FORM DOES NOT EXIST
<i>Meaning</i>	A custom input or output form was specified, but does not exist in the FOENQnnn forms file.
<i>Action</i>	Pre-define the required forms.
<i>Message</i>	10 INVALID INPUT FOR NUMERIC FIELD
<i>Meaning</i>	A value other than a number or valid operator was entered as the search value for a numeric field (item data type E, I, J, K, P, R, or Z).
<i>Action</i>	You may enter only a number and optionally the operators >, >=, <, <=, or : in this field.
<i>Message</i>	11 REQUIRED FIELD(S) NOT DEFINED IN CUSTOM FORM
<i>Meaning</i>	The fields ENQFILE, ENTRIES, and LIMIT are required for this search profile and have not been specified in the output form.
<i>Action</i>	Include the required fields in the form--refer to the <i>Customizing ENQUIRE</i> section.
<i>Message</i>	12 SERIAL READ IN PROGRESS - PLEASE WAIT
<i>Meaning</i>	No SI-path or IMAGE path could be used to access the dataset, so a sequential search is being performed.
<i>Action</i>	Status message only.
<i>Message</i>	13 INVALID OUTPUT FORMAT SPECIFIED
<i>Meaning</i>	You have specified an invalid output format; valid output formats are 1 - 6.
<i>Action</i>	Specify a valid output format, or leave blank.
<i>Message</i>	14 OUTPUT FILE NAME NOT SPECIFIED
<i>Meaning</i>	You press the F4 key to store the found entries but did not specify a file name.
<i>Action</i>	Enter a file name in the Output filename field and press F4 again.
<i>Message</i>	15 OUTPUT FILE ALREADY EXISTS; PRESS F5 AGAIN TO PURGE
<i>Meaning</i>	The specified file to store entries already exists.
<i>Action</i>	Either enter the name of a nonexistent file, suffix the file name with /A to append to the existing file, or press F5 again to purge the existing file.
<i>Message</i>	16 OUTPUT FILE BEING CREATED
<i>Meaning</i>	Store file in the process of being created.
<i>Action</i>	Status message only.
<i>Message</i>	17 NO DIRECT OR INDIRECT RELATIONSHIP EXISTS
<i>Meaning</i>	You have selected a secondary dataset which has no direct or indirect relationship with the primary set or another secondary set.
<i>Action</i>	Datasets must be ordered such that rules of dependence are satisfied.

ENQUIRE error, exceptional, and status messages (cont'd)

<i>Type</i>	<i>Message number/description</i>
<i>Message</i>	18 OUTPUT FILE TO APPEND TO DOES NOT EXIST
<i>Meaning</i>	You have specified an output file with /A that does not exist and which therefore cannot be appended to.
<i>Action</i>	Specify a file that exists, or leave off the /A to create a new output file.
<i>Message</i>	19 SERIAL READ REQUIRED; PRESS F1-F7 TO PROCEED OR F8 TO CANCEL
<i>Meaning</i>	A serial read is necessary to perform the search.
<i>Action</i>	Press any function key in the range F1 - F7 to proceed with the serial read, or F8 to abort it.
<i>Message</i>	20 DATABASE PROFILE CANNOT BE EXECUTED
<i>Meaning</i>	You are attempting to execute a database profile.
<i>Action</i>	This operation is not allowed.
<i>Message</i>	21 NO PREVIOUS ENTRY
<i>Meaning</i>	You are currently accessing the first (or only) entry found.
<i>Action</i>	Note the condition and continue.
<i>Message</i>	22 NO NEXT ENTRY
<i>Meaning</i>	You are currently accessing the last (or only) entry found.
<i>Action</i>	Note the condition and continue.
<i>Message</i>	23 INVALID PASSWORD
<i>Meaning</i>	You have specified an invalid password.
<i>Action</i>	Enter to correct password.
<i>Message</i>	24 INPUT DOES NOT CONFORM TO REQUIRED SYNTAX
<i>Meaning</i>	Syntax is illegal.
<i>Action</i>	Use correct syntax.
<i>Message</i>	25 DEFINITION INCOMPLETE
<i>Meaning</i>	SAVE CURRENT function was invoked before minimum required search profile definition was completed.
<i>Action</i>	Complete search profile by modifying it.
<i>Message</i>	26 SEARCH PROFILE ALREADY EXISTS
<i>Meaning</i>	Attempting to copy a search profile under a name that is already in use for a search profile.
<i>Action</i>	Choose an unused name for the search profile copy.
<i>Message</i>	27 INVALID BASE PROFILE
<i>Meaning</i>	Option B selected and Profile name exceeds six characters.
<i>Action</i>	Specify database name as Profile name.
<i>Message</i>	28 FILE EQUATION FAILED
<i>Meaning</i>	:FILE equation was issued for output file, returned FSERROR.
<i>Action</i>	Correct :FILE equation to prevent FSERROR.
<i>Message</i>	29 MULTIPLE VALUES NOT PERMITTED FOR THIS FIELD
<i>Meaning</i>	Multiple criteria are only allowed for field used as a SUPERDEX SI-key.
<i>Action</i>	You may specify only a single value for this field.

ENQUIRE error, exceptional, and status messages (cont'd)

<i>Type</i>	<i>Message number/description</i>
<i>Message</i>	30 ONLY ONE "I" KEY AND TWO "S" KEYS OR ONE "s" KEY MAY BE SELECTED
<i>Meaning</i>	Maximum supported keys have been exceeded.
<i>Action</i>	Restrict number of keys to limit documented in <i>Appendix C</i> .
<i>Message</i>	31 INVALID CHARACTER USED TO MARK FIELD
<i>Meaning</i>	"I" was used to mark access method for a non IMAGE search field or "S" to mark access method for field which is not a SUPERDEX SI-key.
<i>Action</i>	Access method must conform to key type.
<i>Message</i>	32 HELP NOT AVAILABLE FOR THIS SCREEN
<i>Meaning</i>	Online help does not exist for custom forms.
<i>Action</i>	Remove the label for the F1 function key.
<i>Message</i>	33 MICRO INTERFACE NOT AVAILABLE
<i>Meaning</i>	The Micro Interface option was not purchased.
<i>Action</i>	Contact your Bradmark sales representative for information.
<i>Message</i>	34 DATABASE NAMES MUST BE SPECIFIED CONTIGUOUSLY
<i>Meaning</i>	For a search profile that utilizes multiple databases, base names were not specified in contiguous boxes on the definition form.
<i>Action</i>	Configured bases must be in adjacent boxes.

INDEX

- A**
- Access methods
- Concatenated keys 1-8
 - Custom SI-path 1-10
 - Definition 1-6
 - Determining B-1
 - Generic key retrieval 1-9
 - Greater-than key retrieval 1-9
 - Grouped key retrieval 1-9
 - Keyword retrieval 1-9
 - Less-than key retrieval 1-9
 - Multiple bases 1-10
 - Multiple criteria retrieval 1-10
 - Multiple fields 1-10
 - Multiple keys 1-8
 - Multiple sets 1-10
 - Overview 1-8
 - Partial key retrieval 1-9
 - Range retrieval 1-9
 - Sorted sequential retrieval 1-8
 - Super-grouped retrieval 1-10
- Alphanumeric
- Edit mask 3-6
- ASCII
- Output format 4-5
- Asterisk
- Edit mask 3-6
- Automatic download
- Micro interface 5-4
- B**
- BATCH
- Entry point 5-3
- Batch execution 1-7
- Batch interface 5-2
- Creating input file 5-2
 - Entry point 5-3
 - Executing search profile 5-3
- Binary
- Output format 4-5
- Business Basic
- Floating point decimals 6-7
 - HPFD 6-7
 - SIHPFD JCW 6-7
- C**
- Completing
- Database profile 3-8
 - Search profile 4-13
- Concatenated keys 1-8
- CONTROLLED
- Entry point 5-8
- Controlled interface 5-8
- Copying
- Search profile 4-17
- Creating batch input file 5-2
- Credit sign
- Edit mask 3-7
- Custom forms 6-2
- Custom help forms 6-4
- Custom input forms 6-2
- Custom menus 6-2
- Custom output forms 6-3
- Custom SI-path access 1-10
- Customizing ENQUIRE 6-1
- Customizing forms 4-4
- D**
- Database
- Defining 3-3
- Database password
- Defining 3-3, 4-4
- Database profile
- Completing 3-8
 - Defining 3-2
 - Definition 1-5
 - Deleting 3-10
 - Main menu 3-2
 - Modifying 3-9
- Database profile password
- Defining 3-3
- Database profiles 3-1
- Databases
- Defining 4-4
- Dataset
- Defining 3-4, 4-6
- dBASE
- Output format 4-5
- DBENQ 1-5, 2-1
- Database B-1
- Decimal point
- Edit mask 3-6
- Decimal point position
- Defining 3-8
- DEF
- Entry point 1-5, 2-2, 3, 3-9, 4-2, 14, 17

Defining		ENQUIRE functions.....	2-1
Database	3-3	Entry point	
Database password.....	3-3, 4-4	BATCH.....	5-3
Database profile.....	3-2	CONTROLLED.....	5-8
Database profile password	3-3	DEF	1-5, 2-2, 3, 3-9, 4-2, 14, 17
Databases.....	4-4	RENAME	A-1
Dataset	3-4, 4-6	STRUCT.....	A-1
Decimal point position.....	3-8	ERENQ000	2-1
Edit mask.....	3-6	Error catalog	
Headings	3-6	Native language support	6-6
Input field attributes.....	3-5, 4-12	NLS	6-6
Input fields.....	4-7	Error messages	
Output field attributes.....	3-5, 4-12	ENQUIRE.....	D-1
Output fields	4-10	Errors	
Prompts.....	3-6	SIMAINT.....	D-1
Search profile password.....	4-5	Examples	
SI-Path access	4-9	Edit mask.....	3-7
Definition		Exceptional condition messages	
Access methods	1-6	ENQUIRE.....	D-1
Database profile.....	1-5	Executing	
Input format	1-6	Search profile.....	4-19
Native Language Support	1-7	Executing as son process	5-3
NLS.....	1-7	Executing search profile in batch	5-3
Online help	1-7		
Output format	1-6	F	
Screen formats	1-7	Floating point decimals	
Search profile.....	1-4, 5, 4-1	Business Basic.....	6-7
Security.....	1-6	FOENQ000.....	2-1
Deleting		Forms file	
Database profile.....	3-10	Native language support	6-6
Search profile.....	4-16	NLS	6-6
Dollar sign		Forms specification	4-4
Edit mask	3-6	Function keys	2-3
E			
Edit mask		G	
Alphanumeric	3-6	Generic key retrieval.....	1-9
Credit sign	3-7	Greater-than key retrieval	1-9
Defining.....	3-6	Grouped key retrieval	1-9
Examples	3-7		
Negative sign	3-7	H	
Numeric	3-6	Headings	
Editing profile.....	3-4, 4-5	Defining.....	3-6
ENQLIST.....	2-2	HPFD	
ENQUIRE		Business Basic.....	6-7
Custom forms	6-2		
Custom help forms.....	6-4	I	
Custom input forms	6-2	Input field attributes	
Custom output forms	6-3	Defining.....	3-5, 4-12
Customizing.....	6-1	Input fields	
Invoking.....	2-2	Defining.....	4-7
Line mode	5-8		
Redirection	2-1		

Input format
 Definition 1-6
 Installation 2-1
 Invoking ENQUIRE..... 2-2

K

Keyword retrieval..... 1-9

L

Less-than key retrieval 1-9
 Line mode 5-8
 LIST ALL
 Reporting 4-27
 LIST SUMS
 Reporting 4-29
 Lotus interface 5-6
 REFLECTION 5-6

M

MAC
 Output format..... 4-5
 Main menu 3-2, 4-2, 19
 Manual download
 Micro interface..... 5-4
 Maximum limits C-1
 Message catalog
 Native language support..... 6-6
 NLS..... 6-6
 Micro interface 5-4
 Automatic download 5-4
 Manual download 5-4
 REFLECTION 5-5
 Modifying
 Database profile 3-9
 Search profile..... 4-14
 Multiple keys 1-8

N

Native Language Support
 Definition..... 1-7
 Native language support 6-5
 Error catalog 6-6
 Forms file..... 6-6
 Message catalog..... 6-6
 Setting language..... 6-5
 Supported languages 6-6
 Negative sign
 Edit mask 3-7
 NLS..... 6-5
 Definition..... 1-7

Error catalog 6-6
 Forms file..... 6-6
 Message catalog..... 6-6
 Setting language..... 6-5
 Supported languages 6-6
 Number of qualifying entries
 Reporting 4-27

O

Online help
 Definition..... 1-7
 Output field attributes
 Defining 3-5, 4-12
 Output fields
 Defining 4-10
 Output format
 Definition..... 1-6
 Output formats..... 4-4
 ASCII..... 4-5
 Binary 4-5
 dBASE..... 4-5
 MAC 4-5
 Self describing 4-5
 WK1..... 4-5
 Output to print 2-2

P

Partial key retrieval..... 1-9
 Primary dataset 4-7
 PRINT
 Reporting 4-30
 Profile
 Editing 3-4, 4-5
 Saving 3-4, 4-5
 Prompts
 Defining 3-6

R

Range retrieval..... 1-9
 Redirection
 ENQUIRE..... 2-1
 Output to printer 2-2
 REFLECTION
 Execution command file 5-5
 SDX.PRN command file..... 5-6
 Relational Access
 Multiple criteria retrieval 1-10
 Relational access
 Multiple bases 1-10
 Multiple fields..... 1-10
 Multiple sets 1-10

RENAME		
Entry point	A-1	
Reporting	4-27	
Entries to screen	4-27	
LIST ALL	4-27	
LIST SUMS	4-29	
Number of qualifying entries	4-27	
PRINT	4-30	
Printing entries	4-30	
STORE	4-31	
Storing entries	4-31	
Totals only	4-29	
S		
Saving profile	3-4, 4-5	
Screen formats		
Definition	1-7	
SDX.PRN		
REFLECTION command file	5-6	
Search profile		
All values	4-21	
Completing	4-13	
Compound IMAGE items	4-25	
Copying	4-17	
Definition	1-4, 5, 4-1	
Deleting	4-16	
Executing	4-19	
Generic values	4-22	
Greater-than values	4-23	
Less-than values	4-23	
Main menu	4-2, 19	
Modifying	4-14	
Multiple values in single field	4-24	
Negative values	4-25	
Not-equal-to values	4-23	
Partial values	4-22	
Range values	4-23	
Reserved characters	4-24	
Specifying output file	4-25	
Specifying search limit	4-25	
Specifying values	4-20	
Search profile password		
Defining	4-5	
Secondary dataset	4-7	
Security		
Definition	1-6	
Self describing		
output format	4-5	
Setting language		
Native language support	6-5	
NLS	6-5	
SI-path access		
Defining	4-9	
SIHPFD JCW		
Business Basic	6-7	
SIMAIN		
Errors and exceptional conditions	D-1	
Son process	5-3	
Sorted sequential retrieval	1-8	
Specifying		
Output file	4-26	
Search limit	4-26	
Values	4-20	
All values	4-21	
Compound IMAGE items	4-25	
Generic values	4-22	
Greater-than values	4-23	
Less-than values	4-23	
Multiple values in single field	4-24	
Negative values	4-25	
Not-equal-to values	4-23	
Partial values	4-22	
Range values	4-23	
Reserved characters	4-24	
Status messages	D-1	
STORE		
Reporting	4-31	
STRUCT		
Entry point	A-1	
Summing		
Edit mask	3-7	
Super-grouped retrieval	1-10	
Supported languages		
Native languages support	6-6	
NLS	6-6	
W		
WK1		
Output format	4-5	

Manual Correction/Evaluation Form

Please help us out by indicating any errors you have found in this manual and any comments about anything regarding this manual:

ENQUIRE User Manual

Version 4.2

Released May, 1993

Name _____

Company _____

Address _____

Phone _____

Comments _____

Return to:

BRADMARK TECHNOLOGIES, INC.

4265 San Felipe, Suite 800

Houston, TX 77027

ATTN: Documentation